Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Институт радиоэлектроники и информационных технологий — РТФ

Т. М. ЛЫСЕНКО

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЙ В НАСТОЛЬНОЙ РЕЛЯЦИОННОЙ СУБД

Учебно-методическое пособие

Министерство образования и науки Российской Федерации

Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

Т. М. Лысенко

Проектирование и разработка приложений в настольной реляционной СУБД

Учебно-методическое пособие

Рекомендовано методическим советом
Уральского федерального университета для студентов вуза,
обучающихся по направлениям подготовки
09.03.01 — Информатика и вычислительная техника,
09.03.02 — Информационные системы и технологии,
10.03.01 — Информационная безопасность,
11.03.01 — Радиотехника,
11.03.02 — Информационные технологии и системы связи,
11.03.03 — Конструирование и технология электронных средств,
27.03.04 — Управление в технических системах,
29.03.03 — Технология полиграфического и упаковочного производства

Екатеринбург
Издательство Уральского университета
2018

УДК 004.4152:004.65(075.8)
ББК 32.972.34я73+32.973я73
Л88

Рецензенты: заведующий кафедрой информатики ФБГОУ ВО «Уральский государственный горный университет» канд. техн. наук, доц. *А. В. Дружинин*; д-р техн. наук, проф. кафедры менеджмента и управления качеством Института экономики и управления Уральского государственного лесотехнического университета *В. Г. Лабунец*

Научный редактор — канд. техн. наук, доц. *О. Ю. Иванов*

На обложке использовано изображение с сайта https://goo.gl/21pZqi

Лысенко, Т. М.

Л88 Проектирование и разработка приложений в настольной реляционной СУБД : учебно-метод. пособие / Т. М. Лысенко. — Екатеринбург : Изд-во Урал. ун-та, 2018. — 116 с.
ISBN 978-5-7996-2478-1

В учебно-методическом пособии показана роль и практическая значимость реляционной модели данных, описаны основные понятия и объекты реляционной базы данных, сформулировано учебное задание на проектирование и разработку сложного многотабличного настольного приложения. В работе также выполнен этап проектирования БД, установлены правила целостности, которые соблюдаются на глобальном уровне штатными средствами СУБД. Подробно описаны проектирование таблиц приложения и технология создания всех объектов приложения в СУБД MS Access.

Библиогр.: 11 назв. Табл. 11.

УДК 004.4152:004.65(075.8)
ББК 32.972.34я73+32.973я73

ISBN 978-5-7996-2478-1

Введение

Предметом информатики является информационный ресурс, который стал основным ресурсом человечества, главной ценностью современной цивилизации. Информационные технологии являются как новым средством превращения знаний в информационный ресурс общества, так и средством эффективного использования этого ресурса. Современная трактовка понятия информатики сводится к определению ее как науки, включающей все процессы информатизации общества [1, с. 23].

Информационные технологии — это машинизированные способы обработки семантической информации (данных и знаний), которые реализуются посредством автоматизированных информационных систем (АИС). Важнейшая задача АИС — хранение и обработка данных с помощью специализированного программного обеспечения — систем управления базами данных (СУБД). Базы данных позволяют структурировать, систематизировать и организовывать данные для их хранения в базах. База данных (БД) — это совокупность хранимого во внешней памяти ЭВМ большого объема данных, специальным образом организованных и отображающих состояние определенной предметной области [3–7].

Историю технологии баз данных принято отсчитывать с начала 60-х годов прошлого века, когда были предприняты первые попытки создания СУБД. За прошедшие десятилетия возникали и использовались различные подходы к организа-

ции БД. Для их описания и сравнения используется понятие модели данных, предложенное в 1969 г. Эдгаром Коддом и далее детально разработанное Кристофером Дейтом применительно к реляционной модели [3, с. 48].

Функционирование СУБД основано на введении двух уровней организации БД — **логического** (с точки зрения использования данных в прикладных приложениях) и **физического** (с точки зрения хранения данных в памяти ЭВМ). Формальное описание логического уровня называется моделью данных. Именно модель данных является инструментом, с помощью которого разрабатывается стратегия получения любых данных, хранящихся в БД [6].

В теории баз данных принято выделять следующие основные модели данных: иерархическая, сетевая, реляционная, объектно-ориентированная (объединение реляционной модели с сетевой). Каждой модели соответствует своя СУБД. Самыми ранними СУБД являются иерархические и сетевые, которые использовались в течение многих лет, пока не появились современные реляционные СУБД [7].

Данные в иерархической модели представляются в виде совокупности отдельных древовидных структур, в корнях которых стоят идентификаторы объектов, а на последующих ярусах раскрываются свойства этих объектов. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево). Такая модель удобна для работы с иерархически упорядоченными данными, где возможны только односторонние связи: от старших вершин дерева к младшим. При этом все возможные запросы к БД должны быть отражены в структуре дерева.

Сетевая модель представляет данные в виде графов, где теоретически возможны связи всех со всеми. При этом каждый элемент содержит ссылки на другие, связанные с ним элементы.

В ранних СУБД доступ к данным производился на уровне записей. Пользователи этих систем выполняли явную навигацию в БД, используя языки программирования, расширенные функциями СУБД. Пользователи должны были самостоятельно оптимизировать доступ к данным и следить за целостностью данных в БД без какой-либо поддержки СУБД.

Реляционная модель данных является современной моделью данных, следствием формализации иерархической модели, и свое название получила от математического термина «отношение» (лат. relation). Реляционная модель разработана на основе математической теории отношений и имеет фундаментальное теоретическое обоснование — развитый теоретический аппарат реляционного исчисления и реляционной алгебры [4].

Эта модель опирается на такие важнейшие понятия как таблица, отношение, поле, запись, первичный ключ, внешний ключ, связь, целостность данных. Достоинствами реляционной модели являются ее простота и удобство реализации на современных ЭВМ. Важнейшей из них следует считать возможность формирования гибкой схемы базы данных, допускающей настройку при формировании запросов к БД [9].

До появления ПЭВМ практически отсутствовала персонализация рабочей среды — все программное обеспечение хранилось централизованно и использовалось коллективно, в том числе и оборудование. Основными средствами пользовательского интерфейса были устройства считывания перфокарт и перфолент и неинтеллектуальные терминалы.

Появление первых ПЭВМ в 80-х годах прошлого века привел к созданию **первых настольных (персональных) СУБД** и росту их популярности. Такие СУБД не содержали специальных средств, управляющих данными. Все взаимодействие с данными осуществлялось с помощью файловых сервисов операционной системы, однако в составе таких СУБД были

средства, позволяющие создать более или менее комфортный пользовательский интерфейс.

Лидирующее положение среди первых настольных СУБД с 1986 г. приобретает СУБД dBase, которая обладала комфортной по тем временам средой разработки и средствами манипуляции данными. С 2000 г. dBase становится некоммерческим продуктом с доступными исходными текстами. Хранение данных в dBase основано на принципе: одна таблица — один файл с расширением *.dbf. Формат данных dBase являлся открытым, что позволило другим производителям заимствовать формат для создания dBase-подобных СУБД, например, Clipper, FoxBase.

СУБД Paradox была создана в 1985 г. и сразу стала весьма популярной. Однако, в отличие от dBase, формат данных Paradox является закрытым, поэтому для доступа к данным этого формата требовались специальные библиотеки, например, Paradox Engine. Закрытый формат данных позволяет реализовать такие сервисы СУБД, как защита таблиц и отдельных полей паролем, хранение некоторых правил ссылочной целостности в самих таблицах БД.

СУБД MS Access появилась в начале 90-х гг. [12]. Это была первая настольная реляционная СУБД для 16-разрядной версии Windows, ориентированная на пользователей Office, не знакомых с программированием. Все данные, относящиеся к одной БД (таблицы, индексы, правила ссылочной целостности, список пользователей, а также формы, отчеты, макросы и модули), хранятся в одном файле СУБД [8, 11].

Современные настольные СУБД имеют визуальные средства проектирования запросов, форм, отчетов и приложений; предоставляют доступ к данным серверных СУБД; имеют средства публикации данных в Internet и поддерживают создание приложений для редактирования данных с помощью Web-браузеров.

Следующим шагом в развитии персональных (настольных) СУБД было появление их **сетевых многопользовательских версий**, позволяющих обрабатывать данные, находящиеся на сетевом диске, многим пользователям одновременно благодаря наличию механизма блокировок частей файлов. Поскольку обработка данных в таких версиях по-прежнему осуществляется в пользовательском (клиентском) приложении, возникает возможность нарушения ссылочной целостности данных, что приводит к весьма серьезным ошибкам, таким как разрушение индексов и таблиц. Другими недостатками этих версий являются перегрузка сети и снижение производительности приложений.

Следующим этапом развития СУБД для персональных компьютеров стали **серверные СУБД** [7]. Серверные СУБД реализуют принцип централизации хранения и обработки данных на одном выделенном компьютере в сети, где функционирует сервис операционной системы или специальное приложение, называемое *сервером баз данных*. Реально манипулировать файлами БД может только сервер баз данных. Клиентские приложения функционируют на рабочих станциях типа ПЭВМ в сети, являются источниками запросов к базам данных и не имеют доступа к файлам БД. Сервер баз данных выполняет поддержку ссылочной целостности, резервное копирование, обеспечивает авторизованный доступ к данным, протоколирование операций и, конечно, отвечает за выполнение пользовательских запросов на выборку и модификацию данных и метаданных. Важнейшие преимущества серверных СУБД [7]:

- снижение сетевого трафика при выполнении запросов;
- возможность хранения на сервере бизнес-правил (например, правил ссылочной целостности или ограничений на значения данных), что позволяет избежать дублирования кода в клиентских приложениях;

- возможность реализации части кода, связанного с обработкой данных, в виде хранимых процедур сервера, что позволяет еще больше уменьшить код клиентского приложения и снизить требования к рабочим станциям;
- возможность управления пользовательскими привилегиями и правами доступа к различным объектам базы данных, что обеспечивает защиту данных.

В настоящее время реляционные СУБД, как настольные так и серверные, по распространенности и популярности находятся вне конкуренции. Они стали фактическим промышленным стандартом. Кроме того, объединение реляционной модели с сетевой моделью дает новую модель: объектно-ориентированную, которая используется для создания современных крупных баз данных промышленной и непромышленной сферы со сложными структурами данных.

Изучая основы современных информационных технологий, необходимо понимать практическую значимость реляционной модели баз данных, знать и уметь использовать ее основные понятия в будущей профессиональной деятельности.

1. Объекты реляционной настольной СУБД

1.1. Таблицы и связанные с ними основные понятия реляционной БД

1.1.1. Таблицы

Реляционной считается такая БД, в которой все данные представлены для пользователя в виде прямоугольных таблиц данных и операции над данными сводятся к манипуляциям с таблицами [4]. Таблица отражает тип объекта (процесса или явления) реального мира, а каждая ее запись — конкретный объект. Создание базы данных начинается с планирования ее таблиц и связей между ними. Перед созданием таблиц необходимо тщательно проанализировать требования к приложению БД и выяснить, какие именно таблицы нужны. Для удобной дальнейшей работы с приложением важно правильно определиться с именами таблиц. Таблица должна иметь уникальное имя в БД.

Таблица состоит из столбцов, которые называются *полями*, и из строк, которые называются *записями*. Запись также часто называют экземпляром. Запись состоит из значений полей, часто называемых фактом. Последовательность полей в таблице

и последовательность записей несущественна. Таблица должна иметь не менее одного поля, имя которого должно быть уникальным внутри таблицы. У каждого поля есть тип данных. Тип данных поля определяет данные, которые могут в нем храниться. Тип данных поля необходимо указывать непосредственно при создании поля. Таблица может содержать конечное множество полей различных типов, таких как текст, числа, даты, гиперссылки, вложенные файлы. Значения данных в одном и том же поле должны принадлежать к одному и тому же типу данных, доступному для использования в данной СУБД. У таблиц и полей есть свойства, с помощью которых можно управлять их характеристиками и поведением.

Свойствами таблицы прежде всего являются: описание ее назначения, даты создания и изменения, атрибут скрытия в БД, учетная запись — владелец этой таблицы в базе данных. Некоторые свойства таблицы формируются автоматически. Значения других свойств задаются разработчиком в окне свойств в режиме конструктора таблицы. Например, для таблицы можно задать свойство, определяющее режим отображения таблицы по умолчанию, или правило проверки данных в записи: это выражение, которое должно быть истинно при добавлении или изменении любой записи.

Свойство поля относится к определенному полю таблицы и устанавливает одну из его характеристик или аспект поведения. Перечень свойств поля определяется типом данных поля, а настройка любых свойств выполняется в режиме конструктора таблицы в области свойств поля. Например, для поля можно задать его размер, формат, значение по умолчанию, список всех возможных значений поля, правило проверки данных и сообщение об ошибке.

Поля расположены в таблице в соответствии с порядком следования их имен при создании таблицы. В отличие от полей, записи не имеют имен и номеров. Записи в таблице не упоря-

дочены, невозможно выбрать запись по ее позиции в таблице: среди них не существует «первой», «второй», «последней». Количество записей в таблице логически не ограничено.

1.1.2. Требования к реляционной таблице. Первичный ключ

Реляционная БД состоит из реляционных таблиц. Каждая запись в реляционной таблице должна быть уникальной, то есть в таблице не может быть двух записей с полностью совпадающим набором значений ее полей. Каждое значение, содержащееся в поле таблицы, атомарном (то есть не расчленяемом на несколько значений).

В реляционной таблице необходимо наличие одного поля (или набора полей) для уникальной идентификации каждой записи. Такое поле называется первичным ключом (*primary key*). Если первичный ключ состоит из двух и более полей, он называется составным первичным ключом (*composite primary key*). Первичный ключ в любой реляционной таблице обязан содержать уникальные непустые значения для каждой записи. Таким образом, реляционная таблица имеет первичный ключ, который однозначно идентифицирует каждую ее запись, делает ее уникальной.

С ключевыми полями реляционная СУБД работает особым образом:

1) автоматически проверяет записи таблицы на уникальность по ключевым полям;

2) автоматически создает индексы (индексные таблицы) для ключевых полей, которые предназначены для ускорения поиска данных в таблицах БД.

1.1.3. Связь. Внешний ключ. Виды межтабличных связей

Поскольку данные в реляционной БД по разным объектам, процессам или явлениям хранятся в отдельных таблицах, то таблицы необходимо как-то связать, чтобы можно было лег-

ко комбинировать данные из разных таблиц. Такое взаимоотношение между таблицами называется связью (relationship). Связь — это логическое отношение между двумя таблицами, основанное на их общих полях. Связь таблиц является важнейшим элементом реляционной модели данных. Связи между таблицами в БД позволяют согласовать или объединить данные нескольких таблиц с целью создания общего набора данных для отчета или формы.

Связь таблиц в БД поддерживается внешними ключами (*foreign key*). Внешний ключ, так же как и первичный ключ, является важнейшим понятием в реляционной модели данных. Внешний ключ — это поле в одной таблице, предназначенное для создания и поддержания связи с другой таблицей.

Связь между двумя таблицами БД устанавливается путем присваивания значений внешнего ключа одной таблицы значениям первичного ключа другой. Для создания межтабличной связи данные в поле внешнего ключа должны выбираться только из значений соответствующего поля первичного ключа. Поэтому при проектировании таблицы тип данных поля внешнего ключа должен совпадать с типом данных поля первичного ключа. Также важно учесть следующее: если поле первичного ключа СУБД индексирует автоматически, то поле внешнего ключа должно быть индексировано вручную, на этапе проектирования. Для этого необходимо выбрать соответствующее значение в области свойств поля, учитывая, что тип индекса непосредственно влияет на тип межтабличной связи.

Часто в качестве первичного ключа используется произвольное уникальное числовое значение, которое может генерироваться СУБД. Значение первичного ключа не должно меняться, поскольку в БД с несколькими таблицами первичный ключ одной таблицы может использоваться в качестве внешнего ключа в других таблицах. Если первичный ключ изменяется, это изменение необходимо применить ко всем ссылкам

на этот ключ. Благодаря использованию первичного ключа с постоянным значением снижается вероятность нарушения синхронизации с другими таблицами.

Таблица, содержащая первичный ключ, определяющий возможные значения внешнего ключа, называется главной таблицей или master-таблицей. Таблица, содержащая внешний ключ, называется подчиненной таблицей или detail — таблицей.

Существуют следующие типы межтабличных связей:

- «один-к-одному» (1 : 1);
- «один-ко-многим» (1 : M);
- «многие-ко-многим» (M : M).

Связь 1:1 предполагает, что в каждый момент времени одной записи в таблице **A** соответствует не более одной записи в таблице **B**, и наоборот.

Перед созданием связи «один-к-одному» сначала следует рассмотреть возможность объединения данных в одну таблицу и определить ее первичный ключ. Если этот вариант неприемлем, например, по причине возникновения множества пустых полей или по причине разграничения прав доступа к полям широкой таблицы, следует данные этой таблицы распределить между двумя таблицами и задать в каждой таблице один и тот же первичный ключ.

Если тематика и первичные ключи таблиц различаются, следует выбрать любую из таблиц и вставить ее первичный ключ в другую таблицу в качестве внешнего ключа. При этом в свойствах поля внешнего ключа следует выбрать значение индекса, не допускающего совпадения данных.

Связь 1: M предполагает, что в каждый момент времени одной записи в главной таблице **A** соответствует 0,1 или более записей в подчиненной таблице **B**, но каждая запись в таблице **B** связана не более чем с 1 записью в таблице **A**.

Чтобы создать связь «один-ко-многим», следует первичный ключ таблицы A (на стороне «один») добавить в таблицу

В (на стороне «многие») в виде дополнительного поля, которое будет являться внешним ключом.

Связь М: М предполагает, что в каждый момент времени одной записи в таблице **А** соответствует 0,1 или более записей в таблице **В**, и наоборот. Такая связь реализуется за счет создания третьей, связующей, таблицы, в которой связь «многие-ко-многим» разбивается на две связи «один-ко-многим». Первичные ключи двух таблиц вставляются в третью таблицу. В результате в третьей таблице возникает связь М: М.

Группа связанных таблиц называется схемой базы данных (database schema). Межтабличные связи в интерфейсе СУБД определяются в специальном окне схемы данных, в котором наглядно графически отображаются таблицы в БД, перечень полей таблиц, первичные ключи. Первичный ключ в схеме данных обозначается значком ключа рядом с именем поля. Внешний ключ такого значка ключа не имеет.

Схема данных создается в процессе проектирования БД и может быть изменена при необходимости. Связи между таблицами необходимо иметь в виду еще на этапе планирования таблиц. С помощью такого распространенного и удобного интерфейса, как мастер подстановок, можно установить связь между таблицами, создавая подстановку для поля внешнего ключа, если таблица с соответствующим первичным ключом уже существует в БД.

Организация хранения данных в связанных реляционных таблицах обеспечивает следующие преимущества:

1. Согласованность. Поскольку каждый конкретный объект заносится в базу только один раз и в одну таблицу, вероятность появления неоднозначных или несогласованных данных снижается.

2. Эффективность. Хранение данных в одном месте позволяет сэкономить место на диске. Кроме того, данные из таблиц меньшего размера извлекаются быстрее, чем из больших

таблиц. Наконец, если не хранить данные по разным темам в разных таблицах, в них возникнут пустые значения, указывающие отсутствие данных, а также избыточные данные, что может привести к неэффективному использованию места и снижению производительности.

3. Простота. Структуру базы данных легче понять, если данные по различным темам разделены по разным таблицам.

1.1.4. Нормализация — принцип проектирования реляционной БД

При проектировании базы данных важно понимать и знать, какие сведения являются необходимыми для разрабатываемого приложения, как распределять данные по таблицам и столбцам и как таблицы должны быть связаны друг с другом. Это позволит создать настольную базу данных, отвечающую потребностям и позволяющую быстро вносить в нее изменения при необходимости. Настольная БД с правильной структурой обеспечивает пользователю доступ к обновленным и точным сведениям. Поскольку правильная структура важна для выполнения поставленных задач при работе с настольной БД, целесообразно изучить принципы создания баз данных.

Важнейшим принципом является нормализация. Нормализация представляет собой процесс реорганизации исходных данных для проектирования базы данных путем ликвидации повторяющихся групп данных и иных противоречий в хранении данных с целью приведения таблиц к виду, позволяющему осуществлять непротиворечивое и корректное редактирование данных. Иными словами, нормализация — это минимизация количества повторяющихся данных в реляционной базе данных за счет более эффективной структуры таблиц.

Нормализация устраняет избыточность данных, что позволяет снизить объем хранимых данных и избавиться от различных аномалий их изменения.

Теория нормализации основана на концепции нормальных форм. Принято считать, что база данных находится в данной нормальной форме, если она удовлетворяет определенному набору требований. Теоретически существует пять нормальных форм, но на практике обычно используются только первые три формы.

Первая нормальная форма содержит требование, что каждое поле в таблице должно содержать одно значение, а не список значений.

Вторая нормальная форма содержит требование о том, что каждое поле, не входящее в ключ, должно находиться в зависимости от всего ключевого поля, а не от его части. Это требование применимо в том случае, если первичный ключ состоит из нескольких полей.

Третья нормальная форма содержит требование о том, что поля, не являющиеся ключевыми, должны не только зависеть от всего первичного ключа, но и быть независимыми друг от друга. Другими словами, поле, не являющееся ключевым, должно зависеть только от первичного ключа.

Не вникая в суть теории нормализации, можно выполнить проектирование таблиц базы данных, используя следующие простые правила [12]:

- всегда снабжайте таблицы первичными ключами;
- исключайте дублирование данных;
- проверяйте составной внешний ключ связи;
- проверяйте любые оставшиеся группы информации.

При проектировании реляционных таблиц рекомендуется руководствоваться следующими основными принципами:

- информация в таблице не должна дублироваться;
- не должно быть повторений и между таблицами;
- каждая таблица должна содержать информацию только на одну тему.

При разработке полей для каждой реляционной таблицы необходимо помнить:

- каждое поле должно быть связано с темой таблицы;
- не рекомендуется включать в таблицу поля, которые являются результатом вычислений;
- в таблице должна присутствовать вся необходимая информация;
- информацию следует разбивать на наименьшие логические единицы.

При разработке полей таблицы следует учесть особенности использования вычисляемых полей в таблице [11]:

- вычисляемое поле выводит значение, рассчитанное на основе других данных из этой же таблицы;
- данные из других таблиц нельзя использовать в качестве источника вычисляемых данных;
- вычисляемые поля таблицы не поддерживают некоторые выражения.

1.1.5. Обеспечение целостности базы данных

Целостность БД означает, что реляционная база данных содержит полную и непротиворечивую информацию, необходимую и достаточную для корректного функционирования приложения. Обеспечение целостности является важнейшей характеристикой реляционной СУБД. Реализация целостности подразумевает наличие штатных (встроенных) средств СУБД, позволяющих обеспечить полноту, корректность и непротиворечивость информации в БД в любой момент времени, независимо от того, каким образом данные заносятся в память (в интерактивном режиме, посредством импорта или с помощью программы).

Для работы штатных средств СУБД должны быть установлены правила целостности, которые хранятся в БД и соблюдаются на глобальном уровне:

- отсутствие повторяющихся записей;
- ссылочная целостность;
- ограничения диапазонов возможных значений полей.

СУБД контролирует появление повторяющихся записей, используя встроенные средства контроля уникальности первичных ключей. Первичный ключ любой реляционной таблицы должен содержать уникальные непустые значения. Некоторые (но далеко не все) СУБД могут контролировать уникальность первичных ключей. Если СУБД контролирует уникальность первичных ключей, то при попытке присвоить ключу значение, уже имеющееся в другой записи этой же таблицы, СУБД сгенерирует диагностическое сообщение, обычно содержащее словосочетание *primary key violation*. Это сообщение может быть передано в приложение, с помощью которого пользователь манипулирует данными.

Многие СУБД имеют встроенные средства для назначения первичного ключа, включая средства для работы со специальным типом числовых полей с автоматическим приращением. В случае Microsoft SQL Server такие поля называются Identity fields, в Microsoft Access — поля типа Счетчик, а в случае Corel Paradox — автоинкрементными полями (Autoincrement fields). При добавлении записей такие поля автоматически заполняются случайными или последовательными целыми числами, которые генерируются самой СУБД. СУБД автоматически присваивает новое уникальное значение первичному ключу.

Средства поддержания ссылочной целостности СУБД обеспечивают корректность межтабличных связей и автоматически пресекают любую операцию, приводящую к нарушению ссылочной целостности. Целостность по ссылкам обеспечивается набором правил, поддерживающих установленные межтабличные связи при вводе, редактировании или удалении записей. Если на связи наложены условия целостности данных,

то реляционная СУБД автоматически не позволяет выполнять следующие действия:

- добавлять в связанную таблицу записи, для которых нет соответствующих записей в главной таблице;
- изменять записи в главной таблице таким образом, что после этого в связанной таблице появятся записи, не имеющие соответствующих главных записей;
- удалять записи в главной таблице, для которых имеются подчиненные записи в связанной таблице.

Такие СУБД имеют встроенные средства контроля корректности значений внешних ключей. Если СУБД контролирует корректность значений внешних ключей, то при каждой попытке нарушения ссылочной целостности СУБД сгенерирует диагностическое сообщение, обычно содержащее словосочетание *foreign key violation*, которое может быть передано в приложение.

Большинство реляционных СУБД содержит встроенные штатные средства, контролирующие ограничения диапазонов возможных значений полей. В этом случае в БД должны быть созданы специальные объекты, называемые ограничениями (constraints) или правилами (rules). Эти объекты содержат сведения об ограничениях, накладываемых на возможные значения полей или записей.

Например, с помощью такого объекта можно установить перечень возможных значений данного поля, или его максимальное или минимальное значение, используя свойства поля. После этого СУБД не позволит сохранить в этом пока значение, не удовлетворяющее данному ограничению.

В окне свойств таблицы можно задать правило проверки данных при добавлении или изменении любой записи, введя выражение, операндами которого могут быть поля данной записи. После этого СУБД не позволит сохранить в БД запись, не удовлетворяющую данному правилу.

1.1.6. Индексирование

Поиск или сортировка данных в БД являются операциями, наиболее часто выполняемыми пользователями. Таблицы в БД могут иметь большое количество записей, которые хранятся в произвольном порядке, поэтому поиск данных в БД по заданному критерию отбора путем последовательного просмотра записей таблицы может занимать много времени.

Реляционная СУБД для выполнения быстрого поиска или сортировки данных использует штатные средства — индексы (или индексные таблицы). Известно, что записи в реляционных таблицах неупорядоченные, но любая запись в конкретный момент времени имеет вполне определенное физическое местоположение в файле базы данных. Местоположение записи в файле может изменяться в процессе редактирования данных пользователями, а также в результате манипуляций с файлами БД, проводимых самой СУБД. Местоположение записи в файле задается соответствующим указателем на эту запись (адресом).

Индексирование — это способ реляционной базы данных предварительно сортировать записи одной таблицы в разнообразных порядках, реализуемых одновременно. При индексировании таблицы в базе данных создаются индексные (служебные) таблицы, каждая из которых содержит отсортированные в заданном порядке записи исходной таблицы. Индексная таблица формируется из значений одного поля (индекс простой) или нескольких полей таблицы (индекс составной) и указателей (адресов) на соответствующие записи исходной таблицы.

Индексные таблицы увеличивают объем базы данных, но реально ускоряют поиск. Индекс позволяет быстро находить нужную запись по заданному значению. Ускорение работы с использованием индексов достигается в первую очередь

за счет того, что индекс имеет структуру, оптимизированную под поиск. При использовании индексов в реляционной СУБД поиск записи осуществляется путем предварительного просмотра индекса. В результате поиска в индексной таблице сокращаются объем просматриваемой памяти, занимаемой записями исходной таблицы, и время поиска.

После нахождения нужного указателя (или указателей) в индексной таблице СУБД формирует результаты поиска, обращаясь уже к исходной таблице БД. При проектировании индексов в БД следует учитывать следующее:

- наличие индексов увеличивает объем БД примерно в два раза, но за счет этого существенно ускоряется поиск данных в БД;
- при вводе или редактировании данных таблиц выполняется соответствующая сортировка записей в индексных таблицах, что замедляет операции ввода или редактирования;
- рекомендуется создавать индексы в БД тогда, когда БД уже заполнена;
- индексировать следует те поля, по которым наиболее часто выполняется поиск или сортировка данных.

Перед созданием индекса необходимо решить, следует создать индекс для одного поля или для нескольких полей (составной индекс). Индекс для одного поля создается с помощью установки соответствующего свойства поля. В некоторых СУБД это свойство может принимать значение: совпадения не допускаются, что приведет к созданию уникального индекса для этого поля. Как уже отмечалось выше, такой индекс используется для создания межтабличной связи «один-к одному».

Если предполагается, что часто будет выполняться поиск или сортировка по нескольким полям, можно создать составной индекс для этого сочетания полей. Составной индекс со-

здается в окне конструктора таблицы. В этот индекс можно включить ограниченное количество полей, которое зависит от СУБД. Последовательность полей определяется при создании составного индекса. При сортировке таблицы по составному индексу СУБД сначала выполняет сортировку по первому полю, затем выполняется сортировка по второму полю, заданному для индекса, и т. д. По умолчанию устанавливается сортировка по возрастанию.

Первичный ключ, в том числе составной, индексируются СУБД автоматически, при этом СУБД устанавливает уникальное значение индекса. Если индекс становится ненужным или приводит к значительному снижению производительности, его можно удалить. При этом удаляется только сам индекс, а не поля, на которых он основан.

1.2. Запросы

Для просмотра, добавления, изменения или удаления данных из базы данных удобно использовать запросы. Кроме того, запросы позволяют автоматизировать выполнение многих задач управления данными и просматривать изменения данных перед их фиксацией в БД. С помощью запросов можно получить ответы на очень специфические вопросы о данных, ответить на которые, просто посмотрев на данные в таблице, было бы непросто. Запросы можно использовать для фильтрации данных, выполнения расчетов на основе данных и отображения сводных данных.

Запрос — требование на отбор данных, хранящихся в таблицах, или требование на выполнение определенных действий с данными. Запрос позволяет создать общий набор записей из данных, находящихся в разных таблицах, и использовать этот набор как источник данных для формы или

отчета. Для вывода результата расчета выражения в запросе определяется вычисляемое поле, значение которого пересчитывается при каждом изменении данных в выражении. Существует несколько типов запросов:

- запрос на выборку — средство отбора данных из одной или нескольких таблиц при помощи определенного пользователем условия. Запросы позволяют создавать виртуальные таблицы, которые состоят из вычисляемых полей или полей, взятых из других таблиц. После вывода результатов запроса на экран становится возможным их просмотр, а в некоторых случаях — изменение данных в базовых таблицах. В отличие от этого типа запроса, в запросах на изменение производится изменение данных;
- запрос на изменение — запрос, в котором выполняется копирование или изменение данных. Существуют следующие типы запросов на изменение: запросы на добавление записей, на удаление записей, на создание таблицы и на обновление[1]:

  1) запрос на добавление записей — запрос на изменение, в котором отобранный в запросе результирующий набор записей добавляется в конец таблицы.

  2) запрос на удаление записей — запрос на изменение, в котором удаляются записи, удовлетворяющие указанному условию отбора. Запрос на удаление, что позволяет просмотреть удаляемые строки перед выполнением удалением.

  3) запрос на создание таблицы — запрос на изменение, в котором на основе результирующего набора записей запроса создается новая таблица.

[1] Запрос на изменение невозможно отменить, поэтому перед обновлением следует создать резервные копии всех таблиц, которые будут обновлены запросом на обновление.

4) запрос на обновление — запрос, в котором изменяются значения записей, удовлетворяющих указанному условию отбора.

5) запрос итоговый — запрос, в котором выводятся результаты статистических расчетов, например, среднего или суммы значений различных полей из одной или нескольких таблиц. Строго говоря, итоговые запросы не образуют самостоятельный тип запросов, скорее, это запросы на выборку с расширенными функциональными возможностями.

С помощью запросов (Query) выполняется модификация и выборка данных, изменение метаданных и некоторые другие операции. Большинство СУБД содержат визуальные средства для создания запросов.

Structured Query Language (SQL) — это непроцедурный язык, используемый для формулировки запросов к базам данных и управления данными в большинстве современных СУБД и в настоящее время являющийся индустриальным стандартом. Язык SQL был создан в начале 70-х годов в результате исследовательского проекта IBM, целью которого было создание языка манипуляции реляционными данными [2].

Непроцедурность языка означает, что на нем можно указать, что нужно сделать с базой данных, но нельзя описать алгоритм этого процесса. Иными словами, в этом языке отсутствуют алгоритмические конструкции, такие как метки, операторы цикла, условные переходы и др. Все алгоритмы обработки SQL-запросов генерируются самой СУБД и не зависят от пользователя.

Визуальное построение запроса в интерфейсе СУБД приводит к генерации его текста на языке SQL. Запрос можно написать непосредственно на языке SQL. Понимание принципов работы SQL помогает создавать улучшенные запросы и упрощает исправление запросов, которые возвращают неправильные результаты.

Язык SQL используется для управления данными, их модификации, добавления и удаления данных в имеющихся объектах базы данных. Выбор данных представляет собой наиболее часто встречающуюся операцию, выполняемую с помощью SQL. Оператор SELECT — один из самых важных операторов этого языка, применяемый для выбора данных. Синтаксис этого оператора имеет следующий вид:

- *SELECT column-list;*
- *FROM table-list;*
- *[WHERE where-clause];*
- *[ORDER BY order-by-clause].*

Операторы SELECT должны содержать слова SELECT и FROM; другие ключевые слова, такие как WHERE или ORDER BY, являются необязательными. За ключевым словом SELECT следуют сведения о том, какие именно поля необходимо включить в результирующий набор данных. Знак (*) обозначает все поля таблицы. Для указания имен таблиц, из которых выбираются записи, применяется ключевое слово FROM.

Для фильтрации результатов, возвращаемых оператором SELECT, можно использовать предложение WHERE, синтаксис которого имеет вид:

WHERE expression1 [{AND | OR} expression2 [:]].

Предложение ORDER BY (необязательное) применяется для сортировки результирующего набора данных по одному или нескольким полям. Синтаксис предложения ORDER BY имеет вид:

ORDER BY column1 [{ASC | DESC}] [, column2 [{ASC | DESC}] [,:].

Для определения порядка сортировки используются ключевые слова ASC (по возрастанию) или DESC (по убыванию).

Очень доступно об операторах языка SQL написано в популярной книге Мартина Грубера [2], в которой приведено много примеров, понятных даже начинающему разработчику.

1.3. Формы

Известно, что ввод данных непосредственно в таблицы БД является весьма утомительным занятием, что приводит к ошибкам. Поэтому операторам БД нельзя предоставлять прямой доступ к таблицам. Использование форм, которые, в частности, могут быть аналогами бумажных документов, уменьшает вероятность ошибки и снижает утомляемость оператора. Кроме того, каждый оператор БД должен иметь соответствующие права доступа к информации, хранящейся в таблицах. Формы позволяют создать пользовательский интерфейс для работы с таблицами БД с учетом прав доступа.

Форма — это объект БД, который можно использовать для создания интерфейса пользователя для приложения базы данных. Форма может быть напрямую соединена с источником данных, например, с таблицей или запросом, и может использоваться для ввода, изменения или отображения данных из источника данных. Форма может быть и не связанной напрямую с источником данных, но все равно содержать кнопки, надписи и другие элементы управления, необходимые для работы приложения.

Связанные с источником данных формы можно использовать для управления доступом к данным: с их помощью определяется, какие поля или строки данных будут отображаться. Например, некоторым пользователям достаточно видеть лишь несколько полей большой таблицы. Если предоставить им форму, содержащую только нужные им поля, это облегчит им работу с БД. Для автоматизации часто выполняемых действий в форму можно добавить кнопки и другие функциональные элементы. Эти формы можно рассматривать как окна, через которые пользователи могут видеть и изменять базу данных. Рационально построенная форма ускоряет работу с БД, поскольку пользователям не требуется искать то, что им нужно. Внешне привлекательная форма делает работу

с БД более приятной и эффективной, кроме того, она может помочь в предотвращении неверного ввода данных.

Элементы управления являются частью формы или отчета и используются для ввода, изменения или отображения данных. Например, текстовое поле часто используется для отображения данных в отчетах и для ввода и отображения данных в формах. К другим популярным элементам управления относятся кнопки, флажки и поля со списком (раскрывающиеся списки), а также подчиненная форма.

Элементы управления в формах и отчетах могут быть связанными, свободными и вычисляемыми.

Связанные элементы управления — элементы, источником данных которых является поле таблицы или запроса. Связанный элемент управления служит для отображения значений полей базы данных. Значения могут быть текстовыми, числовыми, логическими, датами, рисунками или диаграммами.

Свободные элементы управления — элементы, не имеющие источника данных, и используются для вывода на экран сведений, рисунков, линий или прямоугольников. Примером свободного элемента является надпись, которая отображает заголовок формы.

Вычисляемые элементы управления — элементы управления, источником данных которых является выражение, а не поле. Для задания значения, которое должно содержаться в таком элементе управления, необходимо сформировать выражение, служащее источником данных элемента. Выражение — это сочетание арифметических и логических операторов, имен других элементов управления, имен полей, функций, констант.

1.4. Отчеты

Отчет является конечным продуктом большинства баз данных, т. е. бумажным документом, данные в котором организованы и отформатированы в соответствие с шаблонами

и требованиями пользователя (разнообразные коммерческие сводки, счета, каталоги, списки и рассылки и т. п.). В отчете комбинируются данные из таблиц, из запросов и даже форм. В отчете нельзя изменять значение исходных данных с помощью элементов управления.

Способы создания форм и отчетов имеют много общего. Однако в отчете нет просмотра в режиме таблицы. Существуют режимы: конструктор, предварительный просмотр, представление отчета. Поскольку СУБД узнает все особенности принтера от операционной системы, то для предварительного просмотра отчета необходимо, чтобы принтер в системе был установлен.

В отчете используется гораздо меньше элементов управления, чем в формах, зато гораздо больше разделов. Это заголовок и примечание отчета, верхний и нижний колонтитулы, область данных.

В отчете может выполняться группировка данных, а также могут выводиться итоговые значения по каждой группе записей. В таком отчете область данных делится на группы данных, каждая из которых имеет **Заголовок группы** и **Примечание группы**. Текст, помещенный в заголовке или примечании группы, будет печататься автоматически для каждой группы записей. Именно в этих областях следует размещать свободные элементы управления для вычисления итоговых значений по группе записей.

1.5. Макросы и модули

Создание базы данных обычно начинается с создания нескольких объектов, например, таблиц, форм и отчетов. Рано или поздно приходится прибегать к программированию, чтобы автоматизировать некоторые процессы и связать

объекты БД. Программированием в СУБД называется процесс расширения функциональности базы данных с использованием макросов или кода Visual Basic for Applications (VBA). Предположим, что пользователь создал форму и отчет и желает добавить в форму кнопку, при нажатии на которую будет открываться отчет. В этом случае программирование будет включать создание макроса или процедуры VBA и настройку свойства события для кнопки, чтобы при ее нажатии запускался макрос или процедура.

Макросы в СУБД рассматриваются в качестве упрощенного языка программирования, на котором можно создавать код, составляя список выполняемых действий из команд, называющихся в некоторых СУБД макрокомандами. Макрокоманда — это основной компонент макроса, самостоятельно или в комбинации с другими макрокомандами определяющий выполняемые в макросе действия.

Для создания макросов в СУБД используется конструктор макросов. При создании макроса каждое действие выбирается в раскрывающемся списке, а затем дополняется необходимыми данными. Макросы позволяют расширять функциональность форм, отчетов и элементов управления без написания кода в модуле VBA. Макросы предоставляют подмножество команд, доступных в VBA. Считается, что создание макросов является более простым, чем написание кода VBA.

Объекты (например, формы и отчеты) и элементы управления (например, кнопки и поля) имеют множество разных свойств событий, с которыми можно связать макросы или процедуры. Каждое свойство события связано с определенным событием, например, щелчком кнопки мыши, открытием формы или изменением данных в текстовом поле. События могут также запускаться внешними по отношению к СУБД факторами, такими как системные события либо макросы или процедуры, связанные с другими событиями. Если

добавить значительное количество макросов или процедур к нескольким свойствам событий многих объектов, БД может стать довольно сложной, но в большинстве случаев можно достичь необходимого результата почти без программирования.

Макросы могут быть автоматически преобразованы в модули VBA или модули классов. Код VBA добавляется в модуль класса формы или отчета. Модуль класса становится частью формы или отчета и перемещается с ними при их копировании или перемещении. Как и макросы, код на языке VBA позволяет автоматизировать процессы и расширять функциональные возможности приложений СУБД. Пользователи могут расширить возможности VBA с помощью элементов управления сторонних разработчиков, а также написать собственные функции и процедуры для своих нужд.

Решение об использовании макросов или кода VBA следует принимать на основе требований к безопасности и функциональности приложения. Проблемы с безопасностью могут возникнуть потому, что на VBA можно написать код, ставящий под угрозу безопасность данных, или повредить файлы на компьютере. При использовании БД, созданной кем-то другим, следует разрешать выполнение кода VBA только в том случае, если БД получена из надежного источника. При создании БД, которая будет использоваться другими пользователями, следует по возможности избегать применения средств программирования, требующих от пользователя выражать доверие БД.

Для обеспечения безопасности базы данных следует по возможности использовать макросы, прибегая к коду VBA для выполнения только тех операций, которые нельзя выполнить с помощью макрокоманд. Такое ограничение использования макрокоманд позволяет пользователям быть уверенными в том, что БД не содержит программных фрагментов, которые могут нанести вред данным и другим файлам на компьютерах.

2. Средства обеспечения безопасности базы данных

Проблема защиты информации является важнейшей в сфере информационных технологий. Информация в настоящее время представляет собой огромные массивы данных, несанкционированный доступ к которым или их разрушение могут приводить к катастрофическим последствиям. Возможность практически мгновенного, без следов, копирования данных, в том числе и удаленно расположенных, провоцирует злоумышленников на несанкционированный доступ к информации, ее несанкционированную модификацию или разрушение.

Как только данные структурированы и сведены в БД, возникает проблема организации доступа к ним некоторого множества пользователей [7]. Очевидно, что нельзя разрешить всем без исключения пользователям беспрепятственный доступ ко всем элементам БД. В любой БД существует конфиденциальная информация, доступ к которой может быть разрешен лишь ограниченному кругу лиц.

Теоретическая проработка вопросов обеспечения безопасности информации и их практическая реализация долгое время отставали от уровня развития СУБД, и в коммерческих версиях СУБД средства обеспечения безопасности данных стали появляться лишь в 90-х годах прошлого века. Исследования по проблемам защиты компьютерной информации опреде-

ляют в качестве составных элементов понятия безопасности информации три компонента:

- конфиденциальность (защита от несанкционированного доступа);
- целостность (защита от несанкционированного изменения информации);
- доступность (защита от несанкционированного удержания информации и ресурсов, защита от разрушения, защита работоспособности).

В основе большинства моделей безопасности баз данных лежит субъектно-объектная модель, в которой выделяются [7]:

- субъекты базы данных (активные сущности: пользователи, коммуникации);
- объекты базы данных (пассивные сущности: таблицы, записи, поля, процедуры, формы, отчеты);
- процессы, порождаемые действиями субъектов БД.

В основе субъектно-объектной модели лежат два принципа безопасности:

- персонализация (идентификация) и аутентификация (подтверждение подлинности) всех субъектов и их процессов по отношению к объектам;
- разграничение полномочий субъектов по отношению к объектам и обязательная проверка полномочий любых процессов над данными.

Простейшая модель безопасности данных строится на основе дискреционного (избирательного) принципа разграничения доступа, при котором доступ к объектам осуществляется на основе множества разрешенных отношений доступа в виде троек «субъект доступа — тип доступа — объект доступа».

Наглядным и распространенным способом формализованного представления дискреционного доступа является матрица доступа, устанавливающая перечень пользователей (субъектов) и перечень разрешенных операций (процессов)

по отношению к каждому объекту базы данных (таблицы, запросы, формы, отчеты). В этом случае важным аспектом моделей безопасности является управление доступом, состоящее из следующих подходов:

- добровольное управление доступом;
- принудительное управление доступом;
- комбинированный способ управления доступом.

При добровольном управлении доступом вводится так называемое владение объектами. Как правило, владельцами объектов являются те субъекты БД, процессы которых создали соответствующие объекты. Добровольное управление доступом заключается в том, что права на доступ к объектам определяют их владельцы. Иначе говоря, ячейки матрицы доступа заполняются теми субъектами (пользователями), которым принадлежат права владения над соответствующими объектами базы данных. В большинстве СУБД права владения объектами могут передаваться.

Принудительное управление доступом предусматривает введение единого централизованного администрирования доступом. В БД выделяется специальный доверенный субъект (администратор), который определяет разрешения на доступ всех остальных субъектов к объектам БД. Иначе говоря, заполнять и изменять ячейки матрицы доступа может только администратор системы.

На практике может применяться комбинированный способ управления доступом, когда определенная часть полномочий на доступ к объектам устанавливается администратором, а другая часть — владельцами объектов.

Комбинированный способ управления доступом реализует во многих СУБД, так называемую защиту на уровне пользователей, когда установление и контроль привилегий пользователей — прерогатива администратора БД. Привилегии — это операции, которые разрешено выполнять пользователю

над конкретными объектами. Администратор БД управляет привилегиями как отдельных физических пользователей, так и групп пользователей. Пользователи и группы создаются и удаляются администратором. Пользователь-владелец данных в БД по собственному усмотрению ограничивает круг пользователей, имеющих доступ к объектам, которыми он владеет, а также управляет привилегиями на свои объекты.

В некоторых настольных реляционных СУБД используется субъектно-объектная модель защиты данных и комбинированный способ управления доступом, который реализует так называемую защиту на уровне пользователей. Основными причинами использования защиты на уровне пользователей являются:

- защита приложения от повреждения из-за неумышленного изменения пользователями таблиц, запросов, форм, отчетов и макросов, от которых зависит работа приложения;
- защита конфиденциальных сведений в БД.

В некоторых СУБД система защиты может быть открытой и закрытой. В открытой системе доступ, если только он не запрещен специально, предоставляется всем пользователям приложения, даже если они не известны системе. В закрытой системе доступ предоставляется только тем пользователям, кому он был назначен. Для защиты данных в таких СУБД используется системная БД, в которой хранятся учетные записи пользователей и групп.

Разрешение на доступ к конкретным объектам сохраняются в файле БД. Существует два типа разрешений: явные и неявные. Разрешения называются явными, если они присвоены учетной записи пользователя; такие разрешения не влияют на разрешения других пользователей. Неявными называются разрешения, присвоенные группе. Пользователь, включенный в группу, получает все разрешения, предостав-

ленные группе. При попытке пользователя выполнить какую-либо операцию с защищенным объектом БД его текущие разрешения определяются комбинацией явных и неявных разрешений на доступ. На уровне пользователей всегда действуют минимальные ограничения из налагаемых явными разрешениями для пользователя и для всех групп, к которым принадлежит данный пользователь.

В настоящее время популярен такой способ защиты данных в БД, как создание списка пользователей (users) с именами (user names) и паролями (passwords).

В этом случае любой объект БД принадлежит конкретному пользователю, и этот пользователь предоставляет другим пользователям разрешение на чтение или модификацию данных из этого объекта либо на модификацию самого объекта. Этот способ применяется во всех серверных и некоторых настольных СУБД. Некоторые СУБД, в основном серверные, поддерживают не только список пользователей, но и роли. Роль — это набор привилегий. Когда пользователь получает одну или несколько ролей, то вместе с ними — и все привилегии, определенные для данной роли.

Любая реляционная СУБД, поддерживающая списки пользователей и ролей, должна их где-то хранить. В дополнение к этим спискам многие СУБД хранят списки таблиц, индексов, триггеров, процедур и др., а также сведения о том, кто ими владеет. Эти списки называются системными таблицами, а соответствующая часть БД называется системным каталогом.

В некоторых версиях СУБД для защиты приложения нельзя создавать учетные записи пользователей и пароли, поскольку средство обеспечения безопасности на уровне пользователя в них не используется. Распространенным средством защиты приложений СУБД является использование стойких алгоритмов шифрования с использованием пароля БД. В процессе шифрования происходит перемешивание данных в таблицах,

что исключает несанкционированный просмотр этих данных. При использовании пароля для шифрования БД все данные становятся не читаемыми в других программах, и для того чтобы использовать эту базу данных, пользователи должны вводить пароль. Пароль БД можно удалить, только если предварительно ввести действующий пароль. Если этот пароль утерян, то БД остается зашифрованной и использовать ее невозможно [11].

3. Проектирование и разработка приложения (на примере СУБД Access «Заработная плата»)

3.1. Достоинства СУБД Access для начинающего разработчика

Задание на проектирование и разработку приложения предлагается выполнять в СУБД MS Access. Access в переводе с английского означает «доступ». MS Access — это функционально полная настольная (персональная) реляционная СУБД, одна из самых мощных, гибких и простых в использовании СУБД. Популярность СУБД Microsoft Access обусловлена несколькими причинами. Access является одной из самых легкодоступных и понятных систем, как для профессионалов, так и для начинающих пользователей [10, 11].

В Access можно разработать один из двух типов БД: БД для настольных компьютеров или веб-базу данных. От выбранного варианта зависит, какие функции будут доступны для работы с БД. БД для настольных компьютеров нельзя опубликовать в Интернете, а веб-базы данных не поддерживают некоторые функции БД для настольных компьютеров, например, итоговые запросы. В соответствии с заданием предполагается разработка БД для настольного компьютера.

Access позволяет быстро и легко разрабатывать приложения реляционных БД, предназначенные для управления информацией. В Access можно разрабатывать БД для отслеживания практически любых сведений, в том числе складских запасов, контактных данных и бизнес-процессов. Приложение Access поставляется с шаблонами БД, которые сразу же можно использовать для работы с различными данными, что позволяет быстро освоить приложение даже начинающим пользователям.

Также понять основные принципы работы с БД помогают полностью русифицированная версия, набор «мастеров» по разработке объектов, облегчающий создание таблиц, форм и отчетов. Благодаря WYSIWIG пользователь видит все результаты своих действий, а идеология Windows позволяет представлять информацию красочно и наглядно.

СУБД Access входит в пакет MS Office [9, 8, 10], поэтому важнейшим достоинством системы является ее интегрированность с Excel, Word и другими программами пакета Office. Данные, созданные в разных приложениях, входящих в этот пакет, легко импортируются и экспортируются из одного приложения в другое. Все составляющие базы данных Access хранятся в едином файле.

Access характеризуется простотой средств разработки и инструментов для управления БД. Access предоставляет весьма удобный интерфейс для создания и эксплуатации достаточно мощных БД без необходимости что-либо программировать. В то же время работа с Access не исключает возможности программирования. Благодаря встроенному языку VBA в Access можно писать приложения, работающие с БД. Основные компоненты Access (построитель таблиц; построители экранных форм, запросов и отчетов) могут вызывать скрипты на языке VBA. Поэтому MS Access позволяет разрабатывать БД и приложения практически «с нуля».

В пособии представлена самостоятельная разработка настольного приложения в СУБД MS Access «Заработная пла-

та сотрудников образовательного учреждения» (см. п. 3.3) на основе исходных данных к проектированию. База данных приложения должна отвечать требованиям нормализации, целостности и непротиворечивости данных, а пользовательский интерфейс должен быть удобным и дружелюбным.

В подразделе ниже приводятся материалы по проектированию и разработке приложения. Приложение предлагается разработать в СУБД MS Access версии 2010 на основе анализа исходных данных и требований к объектам БД, приведенных ниже.

3.2. Интерфейс Access

Прежде чем начать разработку собственного приложения, следует ознакомиться с интерфейсом Microsoft Office Access. Начать работу в СУБД Access достаточно просто: нужно лишь запустить приложение и на вкладке **Файл** выбрать команду для создания новой БД. Приступая к работе в Access версии 2010 и позднее, рекомендуется ознакомиться с конкретными функциями и возможностями, доступными в этой версии. Справочная система приложения вызывается клавишей **F1** или кнопкой (**?**) в правом верхнем углу окна приложения. Раскрывая последовательно нужные пункты вкладки **Оглавление**, или используя панель **Поиск**, можно получить самую достоверную и полную информацию по интересующему вопросу, включая актуальную справку с сайта разработчика [11].

Для получения справки в открытом окне диалога следует использовать кнопку со знаком вопроса в верхнем правом углу окна, что позволит открыть справку Access непосредственно по параметрам данного окна.

Окно СУБД Access является главным рабочим местом для создания и работы с базами данных и состоит из следующих областей.

- ***Строка заголовка приложения***, в правой части которой расположены три *кнопки* для управления окном приложения. В левой части (по умолчанию) находится *панель быстрого доступа*, которая предназначена для быстрого доступа к наиболее часто используемым функциям. Самая левая кнопка — кнопка MS Office (А) — открывает меню с командами для работы с окном приложения. В центре строки заголовка выводится имя активной БД.
- ***Лента***, которая разработана для облегчения доступа к командам, состоит из *вкладок*, связанных с определенными целями или объектами. Каждая вкладка, в свою очередь, состоит из *нескольких групп* взаимосвязанных элементов управления. В каждой *группе* находятся *кнопки команд*, которые служат для выполнения команд или отображения меню команд. В некоторых группах в правом нижнем углу могут отображаться маленькие значки — *кнопки вызова диалоговых окон*. После нажатия такой кнопки открывается соответствующее диалоговое окно или область задач, содержащая дополнительные параметры, связанные с данной группой. В результате *лента* вмещает большой объем содержимого — кнопок, коллекций, элементов диалоговых окон и т. д. С помощью настроек *ленту* можно персонализировать, создавая *настраиваемые вкладки* и группы, содержащие часто используемые кнопки команд. Кроме стандартного *набора вкладок*, которые отображаются на *ленте* при запуске, существуют вкладки еще двух типов, которые появляются в интерфейсе в зависимости от выполняемой задачи: *контекстные вкладки* и *вкладки приложений*.

Контекстные вкладки позволяют работать с объектом БД в режиме конструктора или, например, в режиме таблицы.

Набор контекстных вкладок, выделенный цветом, появляется рядом со стандартными вкладками на ленте.

Вкладки представлений заменяют стандартный набор вкладок при переходе в определенные представления, которые появляются при выборе какой-либо команды на вкладке **Файл**. Вкладки представлений содержат команды и сведения, применимые ко всей БД, например, **Сжать и восстановить**, **Зашифровать паролем**, **Печать**. Вкладка **Файл** открывается при запуске приложения Access, если при этом не открывается БД (например, при запуске приложения Access из меню «Пуск»).

Под лентой Access открывает **Панель сообщений**. На панели сообщений отображаются предупреждения системы безопасности, если в открываемом файле имеется потенциально опасное активное содержимое (такое как макросы, элементы ActiveX, подключения к данным и т. д.). В такой ситуации появляется желтая или красная панель сообщений со значком щита и извещением о возможных проблемах. Если известно, что источник содержимого надежен, то следует нажать на желтой панели сообщений кнопку **Включить содержимое**, чтобы пометить документ как надежный, либо включить содержимое для текущего сеанса. На красной панели сообщений можно щелкнуть текст предупреждения, а затем выбрать команду **Все равно редактировать**.

- ***Окно объекта БД*** обрамлено справа и внизу полосами прокрутки для перемещения по активному открытому объекту БД.
- ***Область навигации*** — область в левой части окна Access, предназначенная для работы с объектами БД. Область навигации позволяет организовать объекты БД и является основным средством их открытия или изменения объектов БД. Область навигации организована по категориям и группам. Пользователи могут

выбрать различные параметры организации, а также создать собственную схему организации. По умолчанию в новой базе данных используется категория типа объекта, которая содержит группы, соответствующие различным типам объектов БД. Область навигации можно уменьшить или скрыть, но она не перекрывается открытыми окнами объектов базы данных.

- ***Строка состояния***, содержимое которой слева зависит от выполняемой операции. Справа в строке состояния выводятся кнопки для быстрого выбора *режима* открытого объекта. Окно для настройки строки состояния открывается щелчком правой клавиши мыши по строке состояния.

Контекстное меню приложения реализовано в следующих областях: строка заголовка приложения, панель быстрого доступа, лента, вкладки, строка состояния, вертикальная полоса прокрутки, окно БД.

Для настройки пользовательского рабочего места выбирается команда **Параметры** на вкладке **Файл.** Эта команда открывает своеобразный командный пункт приложения, в котором пользователь, перемещаясь по категориям настроек, определяет параметры пользовательского интерфейса, параметры для текущей БД, параметры внешнего вида таблиц, параметры конструкторов объектов, параметры управления безопасностью.

Разработчики приложения выделяют три основных компонента пользовательского интерфейса Access 2010: лента, область навигации и набор команд на вкладке **Файл**. Три этих элемента формируют среду, в которой создаются и используются БД.

Откройте приложение **MS Access**. Выполните команду **Файл/Параметры**. Просмотрите и установите подходящие вам параметры приложения по категориям: **Общие, Текущая**

база данных, Таблица, Конструкторы объектов, Правописание, Язык, Параметры клиента.

3.3. Исходные данные

Исходные данные к проектированию и разработке приложения — это вымышленные сведения, которые характеризуют некоторую группу сотрудников образовательного учреждения (ОУ) и позволяют рассчитать зарплату каждому сотруднику с учетом их профессиональной деятельности. Зарплата сотрудника состоит из оклада по должности с учетом доли ставки сотрудника и надбавок к окладу, которые устанавливаются (в процентах от оклада) соответствующими нормативными документами образовательного учреждения. Размер каждой надбавки за единицу работы не должен быть выше некоторого установленного процента. Ниже приведен перечень исходных данных к проектированию приложения:

- фамилия, имя, отчество сотрудника; дата рождения; дата приема на работу;
- должность (профессор, доцент, ст. преподаватель, преподаватель, ассистент); оклад по должности;
- сотрудник может занимать только одну должность; доля ставки по должности, занимаемой сотрудником: 1,5; 1,25; 1; 0,75; 0,5; 0,25;
- ученая степень сотрудника, надбавка зависит от наличия или отсутствия ученой степени (доктор наук, кандидат наук, без степени);
- стаж работы сотрудника, надбавка зависит от стажа работы, который отсчитывается от даты приема на работу и увеличивается с шагом 5 лет;

- работа в составе государственной аттестационной комиссии (ГАК), надбавка зависит от уровня ГАК (бакалавриат, магистратура, специалитет);
- руководство практикой, надбавка зависит от вида практики (учебная, производственная, преддипломная) и от числа студентов по этому виду практики (число студентов по одному виду практики не более 5);
- научные публикации, надбавка зависит от вида публикации (тезисы, доклад, статья) и от количества публикаций каждого вида (количество публикаций одного вида не более 3);
- участие в НИР, надбавка зависит от вида НИР (разработка, прикладная, фундаментальная) и от количества НИР каждого вида (количество НИР одного вида не более 2);
- направление заявки в Роспатент, надбавка зависит от типа заявки (на изобретение, на полезную модель, на программу ЭВМ).

3.4. Требования к объектам БД и последовательности их разработки

1. Определить перечень таблиц, разработать поля таблиц.
2. Задать первичные и внешние ключи в таблицах приложения.
3. Для всех полей внешних ключей в подчиненных таблицах разработать поля подстановок для удобного ввода данных значений внешних ключей.
4. Установить межтабличные связи, обеспечить при этом целостность данных.
5. Задать условия на значения полей (на работу не принимаются лица младше 25 и старше 65 лет; дата приема на ра-

боту не позже текущей даты; оклад по должности не менее 20 000 руб.; размер каждой надбавки за единицу работы не должен быть выше 5 %).

6. Ввести записи в главные таблицы БД, указанные в исходных данных к проектированию.

7. Ввести вымышленные данные в подчиненные таблицы, соответствующие сведениям не менее чем о 10 сотрудниках образовательного учреждения.

8. Создать запросы, которые выводят следующие данные:

- **Кадры**: Код сотрудника, Фамилия И. О. (инициалы вычисляются), должность, доля ставки, дата рождения, возраст, в годах, дата приема на работу, стаж, в годах (вычисляется от даты приема на работу).
- **НадбавкаЗаСтаж**: Код сотрудника, Стаж, в годах, с учетом заданного шага по стажу 5 лет, Надбавка за стаж, % от оклада.
- **НадбавкаЗаПракт**: Код сотрудника, Надбавка за все практики, % от оклада.
- **НадбавкаЗаПубл**: Код сотрудника, Надбавка за все публикации, % от оклада.
- **НадбавкаЗаНИР**: Код сотрудника, Надбавка за все НИР, % от оклада;
- **НадбавкаЗаГАК**: Код сотрудника, Надбавка за ГАК, % от оклада.
- **НадбавкаЗаСтепень**: Код сотрудника, Надбавка за степень, % от оклада.
- **НадбавкаЗаЗаяв**: Код сотрудника, Надбавка за заявки, % от оклада.
- **Зарплаты**: Код сотрудника, Фамилия И. О., Оклад с учетом доли ставки, руб., Каждая из надбавок за все виды работ, % от оклада, Зарплата, руб.

9. Создать на основе запроса **Кадры** отчет **Кадровый состав,** который выводит все поля запроса **Кадры** и следу-

ющие итоги: всего сотрудников, средний возраст, средний стаж.

10. Создать на основе запроса **Зарплаты** отчет **Заработная плата,** который выводит все поля запроса **Зарплаты** и следующие итоги: всего сотрудников, средняя зарплата, сумма зарплат, руб.

11. Создать формы для ввода и редактирования данных в главных таблицах БД (иначе называемых справочниками).

12. Разработать форму **Сотрудники**, в которой можно просматривать и редактировать все данные о каждом сотруднике и добавлять запись о новом сотруднике. В форме использовать: поля со списками, вычисляемые поля (Возраст, Стаж), подчиненные формы.

13. Разработать **Главную кнопочную форму**, состоящую из страниц: Редактирование справочников, Сведения о сотрудниках, Просмотр отчетов. Каждая из страниц открывает следующую страницу для выбора конкретного объекта. На каждой странице должна быть кнопка возврата на предыдущую страницу. Форма должна автоматически запускаться при открытии приложения.

14. Установить параметры запуска приложения.

15. Реализовать шифрование приложения с использованием пароля.

4. Создание таблиц приложения

4.1. Анализ задания на проектирование приложения

Основной принцип проектирования реляционных БД — это принцип нормализации, фактически требующий отсутствия повторяющихся данных в БД за счет эффективной структуры таблиц, позволяющей осуществлять непротиворечивое и корректное хранение и редактирование данных в любой момент времени (см. п. 1.1.4). При проектировании таблиц реляционной БД следует понимать, что информация в таблицах не должна дублироваться; каждая таблица должна содержать информацию только на одну тему, которая должна быть отражена в названии этой таблицы.

На основании изложенных выше принципов проектирования реляционной БД и анализа исходных данных к проектированию приложения следует весьма очевидный вывод: разрабатываемая реляционная БД должна состоять из нескольких реляционных таблиц (табл. 4.1).

Таблица 4.1

Результаты анализа исходных данных к проектированию БД

Исходные данные	Количество таблиц	Название таблиц
Фамилия, имя, отчество сотрудника, дата рождения; дата приема на работу, доля ставки	1	Сотрудники

Окончание табл. 4.1

Исходные данные	Количество таблиц	Название таблиц
Должность (профессор, доцент, старший преподаватель, преподаватель, ассистент); оклад по должности	1	Должности
Ученая степень, надбавка зависит от наличия или отсутствия ученой степени (доктор наук, кандидат наук, без степени)	1	Ученые степени
Стаж работы, надбавка зависит от стажа работы, который отсчитывается от даты приема на работу и увеличивается с шагом 5 лет	1	Стажи
Работа в составе государственной аттестационной комиссии (ГАК), надбавка зависит от уровня ГАК (бакалавры, магистры, специалисты)	2	Уровни ГАК Работа в ГАК
Руководство практикой, надбавка зависит от вида практики (учебная, производственная, преддипломная) и от числа студентов по каждому виду практики	2	Виды практик Руководство практикой
Научные публикации, надбавка зависит от вида публикации (тезисы, доклад, статья) и от количества публикаций каждого вида	2	Виды публикаций Научные публикации
Участие в НИР, надбавка зависит от вида НИР (разработка, прикладная, фундаментальная) и от количества НИР каждого вида	2	Виды НИР Участие в НИР
Направление в Роспатент заявки, надбавка зависит от типа заявки (на изобретение, на полезную модель, на программу ЭВМ).	2	Типы заявок Заявки в Роспатент

4.2. Проектирование таблиц и полей таблиц

После перечня таблиц приложения следует определиться с перечнем полей каждой таблицы. При этом следует помнить, что каждое поле должно быть связано с темой таблицы; в таблице должна присутствовать вся исходная информация, которую следует разбивать на наименьшие логические единицы; не рекомендуется включать в таблицу поля, которые являются результатом вычислений.

Известно, что каждая запись в реляционной таблице должна быть уникальной (см. 1.1.2). Для уникальной идентификации каждой записи в таблице должно быть поле, которое называется первичным ключом (обычно имена таких полей начинаются со слова «Код»). Для того чтобы Access автоматически при добавлении записей присваивал уникальное числовое значение полю первичного ключа, необходимо подключить встроенное средство генерации первичного ключа. Для этого необходимо для поля первичного ключа задать специальный тип данных, имеющийся в СУБД Access — Счетчик.

Использование этого типа данных является удобным и подходит для первичных ключей всех таблиц, за исключением таблицы **Стажи**. В этой таблице следует использовать ручной ввод числового значения в поле **Код стажа**. Это связано с тем, что стаж каждого сотрудника должен быть вычислен в запросе (с учетом принятой сетки стажей, с шагом 5 лет), т. е. поле **Код стажа** должно вычисляться автоматически. Наличие счетчика может привести к ошибке, когда вычисленного в запросе кода стажа не окажется в таблице **Стажи**.

Результат проектирования полей таблиц приведен ниже (табл. 4.2).

Таблица 4.2

Описание таблиц БД, перечень полей

Таблица	Описание	Перечень полей
Сотрудники	Таблица содержит список сотрудников образовательного учреждения с обязательным указанием следующих данных: фамилия, имя, отчество сотрудника, дата рождения; дата приема на работу, доля ставки	Код сотрудника Фамилия Имя Отчество Дата рождения Дата приема Доля ставки
Должности	Таблица содержит перечень штатных должностей сотрудников с обязательным указанием оклада по каждой должности, в рублях	Код должности Должность Оклад
Ученые степени	Таблица содержит перечень ученых степеней с обязательным указанием надбавки за каждую ученую степень, в % от оклада по должности	Код степени Ученая степень НадбавкаСтепень
Стажи	Таблица содержит сетку стажей с шагом 5 лет: < 5, >= 5, >= 10, >= 15, >= 20, >= 25 … с обязательным указанием надбавки за стаж в этой сетке, в % от оклада по должности	Код стажа Стаж НадбавкаСтаж
Уровни ГАК	Таблица содержит перечень уровней государственной аттестационной комиссии (ГАК) с обязательным указанием надбавки за работу в ГАК, в % от оклада по должности	Код ГАК Уровень ГАК НадбавкаГАК
Виды практик	Таблица содержит перечень видов практик с обязательным указанием надбавки за руководство практикой одного студента в % от оклада по должности	Код практики Вид практики НадбавкаПрактика

Окончание табл. 4.2

Таблица	Описание	Перечень полей
Руководство практикой	Таблица содержит сведения о руководстве практиками сотрудниками с обязательным указанием вида практики и числа студентов по этому виду практики	Код сотрудника Код практики Число студентов
Виды публикаций	Таблица содержит перечень видов научных публикаций с обязательным указанием надбавки за одну публикацию, в % от оклада по должности	Код публикации Вид публикации НадбавкаПубл
Научные публикации	Таблица содержит сведения о научных публикациях сотрудников с обязательным указанием вида публикации и числа публикаций по этому виду	Код сотрудника Код публикации Число публикаций
Виды НИР	Таблица содержит перечень видов научно-исследовательских работ (НИР) с обязательным указанием надбавки за одну НИР, в % от оклада по должности	Код НИР Вид НИР НадбавкаНИР
Участие в НИР	Таблица содержит сведения об участии сотрудников в НИР с обязательным указанием вида НИР и количества НИР	Код сотрудника Код НИР Число НИР
Типы заявок	Таблица содержит перечень типов заявок в Роспатент, с обязательным указанием надбавки за тип заявки, в % от оклада по должности	Код заявки Тип заявки НадбавкаЗаявка
Заявки в Роспатент	Таблица содержит сведения о направлении заявок в Роспатент сотрудниками ОУ с обязательным указанием типа заявки	Код сотрудника Код заявки
Работа в ГАК	Таблица содержит сведения о работе в ГАК сотрудников ОУ с обязательным указанием уровня ГАК	Код сотрудника Код ГАК

После формирования списка таблиц приложения и перечня полей каждой таблицы следует указать тип данных каждого поля (т. е. значение, которое может храниться в поле). Кроме того, на этом этапе анализа во всех таблицах БД следует задать первичный ключ (табл. 4.3). В таблицах **Руководство практикой, Участие в НИР**, **Научные публикации, Заявки в Роспатент, Работа в ГАК** следует предусмотреть составной первичный ключ, который запретит ввод повторяющихся записей по виду практики, виду НИР, виду научной публикации, типу заявки, уровню ГАК для одного и того же сотрудника. Повторяющихся записей в этих таблицах быть не должно, поскольку в трех из них предусмотрено числовое поле для ввода количества студентов, количества НИР, количества публикаций. Аналогичного поля в таблицах **Заявки в Роспатент, Работа в ГАК** не предусмотрено в соответствии с исходными данными.

Таблица 4.3

Типы данных и первичные ключи таблиц

Таблица	Имя поля	Тип данных	Примечание
Сотрудники	Код сотрудника	Счетчик	Первичный ключ
	Фамилия	Текстовый	
	Имя	Текстовый	
	Отчество	Текстовый	
	Дата рождения	Дата/время	
	Дата приема	Дата/время	
	Доля ставки	Числовой	

Продолжение табл. 4.3

Таблица	Имя поля	Тип данных	Примечание
Должности	Код должности	Счетчик	Первичный ключ
	Должность	Текстовый	
	Оклад	Денежный	
Ученые степени	Код степени	Счетчик	Первичный ключ
	Ученая степень	Текстовый	
	НадбавкаСтепень	Числовой	
Стажи	Код стажа	Числовой	Первичный ключ
	Стаж	Текстовый	
	НадбавкаСтаж	Числовой	
Уровни ГАК	Код ГАК	Счетчик	Первичный ключ
	Уровень ГАК	Текстовый	
	НадбавкаГАК	Числовой	
Виды практик	Код практики	Счетчик	Первичный ключ
	Вид практики	Текстовый	
	НадбавкаПрактика	Числовой	
Руководство практикой	Код сотрудника	Числовой	Первичный ключ, составной
	Код практики	Числовой	
	Число студентов	Числовой	

Окончание табл. 4.3

Таблица	Имя поля	Тип данных	Примечание
Виды публикаций	Код публикации	Счетчик	Первичный ключ
	Вид публикации	Текстовый	
	НадбавкаПубл	Числовой	
Научные публикации	Код сотрудника	Числовой	Первичный ключ, составной
	Код публикации	Числовой	
	Число публикаций	Числовой	
Виды НИР	Код НИР	Счетчик	Первичный ключ
	Вид НИР	Текстовый	
	НадбавкаНИР	Числовой	
Участие в НИР	Код сотрудника	Числовой	Первичный ключ, составной
	Код НИР	Числовой	
	Число НИР	Числовой	
Типы заявок	Код заявки	Счетчик	Первичный ключ
	Тип заявки	Текстовый	
	НадбавкаЗаявка	Числовой	
Заявки в Роспатент	Код сотрудника	Числовой	Первичный ключ, составной
	Код заявки	Числовой	
Работа в ГАК	Код сотрудника	Числовой	Первичный ключ, составной
	Код ГАК	Числовой	

4.3. Проектирование связей таблиц, определение внешних ключей

В подразделе 4.1 (табл. 4.1) показано, что исходные данные к проектированию приложения должны храниться в отдельных реляционных таблицах, которые необходимо связать. Связи между таблицами позволяют, прежде всего, обеспечить целостность и непротиворечивость данных в БД в каждый момент времени. Кроме того, межтабличные связи позволяют создать общий набор данных для запросов, форм и отчетов.

Связь между двумя таблицами устанавливается путем присваивания значений внешнего ключа одной таблицы значениям первичного ключа другой. Поскольку первичные ключи таблиц уже определены выше (табл. 4.3), следует понять, какие пары таблиц приложения должны быть связаны, и после этого добавить в связанную таблицу (подчиненную) поле внешнего ключа, имя и тип данных которого совпадает с первичным ключом главной таблицы.

Из исходных данных следует, что заработная плата каждого сотрудника состоит из оклада по должности с учетом доли занимаемой ставки и надбавок к окладу, которые устанавливаются соответствующими нормативными документами образовательного учреждения в процентах от оклада. Поэтому логично принять за главные таблицы в связи «один-ко-многим» те таблицы, в которых содержатся перечень штатных должностей образовательного учреждения и все виды действующих в учреждении надбавок.

В таблице ниже (табл. 4.4) определен перечень связанных таблиц приложения и указаны внешние ключи для создания каждой из связей. Для обеспечения целостности данных для создания связи «один-ко-многим» поле внешнего ключа должно быть индексировано с параметром: совпадения допускаются.

Таким образом, анализ исходных данных к проектированию приложения показал, что разрабатываемое приложение должно состоять из 14 таблиц, между которыми должно быть установлено 12 связей типа «один-ко-многим».

Таблица 4.4

Проектирование связей «один ко многим»

Главная таблица	Подчиненная таблица	Внешний ключ	Тип данных
Должности	Сотрудники	Код должности	Числовой
Ученые степени	Сотрудники	Код степени	Числовой
Уровни ГАК	Работа в ГАК	Код ГАК	Числовой
Виды практик	Руководство практикой	Код практики	Числовой
Виды публикаций	Научные публикации	Код публикации	Числовой
Виды НИР	Участие в НИР	Код НИР	Числовой
Типы заявок	Заявки в Роспатент	Код заявки	Числовой
Сотрудники	Руководство практикой	Код сотрудника	Числовой
Сотрудники	Научные публикации	Код сотрудника	Числовой
Сотрудники	Участие в НИР	Код сотрудника	Числовой
Сотрудники	Заявки в Роспатент	Код сотрудника	Числовой
Сотрудники	Работа в ГАК	Код сотрудника	Числовой

4.4. Проектирование правил проверки

Анализ исходных данных к проектированию приложения показывает, что для корректного функционирования приложения необходимо использовать встроенные штатные средст-

ва СУБД, контролирующие ограничения диапазонов возможных значений полей или записей. Условия на значения полей таблиц проверяются СУБД сразу после ввода данных в это поле и перехода к другому полю. Условия на значения записей позволяют сравнивать значения в двух или нескольких полях одной записи и проверяются после перехода к другой записи.

Ниже перечислены все ограничения, указанные в задании на проектирование приложения (см. п. 3.3, 3.4):

- на работу в ОУ не принимаются лица младше 25 и старше 65 лет;
- дата приема на работу не позже текущей даты;
- оклад по должности не менее 20 000 руб.;
- доля ставки, занимаемой сотрудником: 1,5; 1,25; 1; 0,75; 0,5; 0,25;
- количество студентов по одному виду практики не более 5;
- количество научных публикаций одного вида не более 3;
- количество НИР одного вида не более 2;
- размер любой надбавки за единицу работы не более 5 %.

Результаты проектирования правил проверки значений полей приведены ниже (табл. 4.5).

Таблица 4.5

Проектирование правил проверки значений полей

Таблица	Поле	Условие на значение	Сообщение об ошибке
Сотрудники	Дата рождения	Значение должно находиться в диапазоне от ([Текущая дата] минус 25 лет) до ([Текущая дата] минус 65 лет)	Дата рождения не соответствует требованиям: на работу не принимаются лица младше 25 и старше 65 лет

Продолжение табл. 4.5

Таблица	Поле	Условие на значение	Сообщение об ошибке
Сотрудники	Дата приема	Значение должно быть меньше или равно значению [Текущая дата]	Дата приема на работу не позже текущей даты
Должности	Оклад	Больше или равно 20 000 руб.	Оклад по должности не менее 20 000 руб.
Сотрудники	Доля ставки	Значение равно одному из значений: 1,5; 1,25; 1; 0,75; 0,5; 0,25	Доля ставки может иметь одно из следующих значений: 1,5; 1,25; 1; 0,75; 0,5; 0,25
Руководство практикой	Число студентов	Меньше или равно 5	Количество студентов по одному виду практики не более 5
Научные публикации	Число публикаций	Меньше или равно 3	Количество публикаций одного вида не более 3
Участие в НИР	Число НИР	Меньше или равно 2	Количество НИР одного вида не более 2
Стажи	Надбавка-Стаж	Меньше или равно 0,05	Надбавка за один шаг по стажу не может быть выше 5 %
Уровни ГАК	Надбавка-ГАК	Меньше или равно 0,05	Надбавка за работу в ГАК одного уровня не может быть выше 5 %
Виды практик	Надбавка-Практика	Меньше или равно 0,05	Надбавка за руководство практикой одного студента не может быть выше 5 %
Ученые степени	Надбавка-Степень	Меньше или равно 0,05	Надбавка за ученую степень не может быть выше 5 %

Окончание табл. 4.5

Таблица	Поле	Условие на значение	Сообщение об ошибке
Виды публикаций	Надбавка-Публ	Меньше или равно 0,05	Надбавка за одну публикацию не может быть выше 5 %
Виды НИР	Надбавка-НИР	Меньше или равно 0,05	Надбавка за одну НИР не может быть выше 5 %
Типы заявок	Надбавка-Заявка	Меньше или равно 0,05	Надбавка за направление заявки не может быть выше 5 %

Перечисленные в этой таблице условия на значения полей проверяются СУБД сразу после ввода данных в это поле и перехода к другому полю. В случае нарушения условия СУБД формирует сообщение об ошибке.

Из анализа исходных данных к проектированию следует, что для таблицы **Сотрудники** необходимо задать условие на значение записей, в котором сравниваются значения в полях **Дата приема** и **Дата рождения**. Вообще, значения на условие записей позволяют сравнивать значения в двух или нескольких полях одной записи и проверяются после перехода к другой записи. Такие условия задаются в окне **Свойства** таблицы, открытой в режиме конструктора, в поле **Правило проверки** (или **Условие на значение**).

Алгоритм проверки должен быть следующий: дата приема сотрудника на работу должна находиться в диапазоне от «Дата рождения плюс 25 лет» до «Дата рождения плюс 65 лет». Сообщение об ошибке: На работу принимаются сотрудники в возрасте от 25 до 65 лет.

Таким образом, работу по проектированию реляционных таблиц приложения, полей таблиц, межтабличных связей для

обеспечения целостности данных можно считать выполненной. С целью подведения ее итогов, которые должны использоваться непосредственно в процессе разработки приложения в среде СУБД, целесообразно представить сводные результаты этой работы в табличной форме (табл. 4.6, 4.7).

Таблица 4.6

Перечень полей таблицы Сотрудники и их свойств

Имя поля	Тип данных	Свойства поля
Код сотрудника	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
Фамилия	Текстовый	Размер поля: 30; обязательный ввод данных
Имя	Текстовый	Размер поля: 30; обязательный ввод данных
Отчество	Текстовый	Размер поля: 30; обязательный ввод данных
Дата рождения	Дата/время	Обязательный ввод данных; краткий формат даты; условие на значение и сообщение об ошибке (табл. 4.5)
Дата приема	Дата/время	Обязательный ввод данных; краткий формат даты, условие на значение и сообщение об ошибке (табл. 4.5)
Доля ставки	Числовой	Размер: одинарное с плавающей точкой; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5), значение по умолчанию: 1
Код должности	Числовой	Размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются
Код степени	Числовой	Размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются

Таблица 4.7

Перечень полей других таблиц приложения и их свойств

Таблица	Имя поля	Тип данных	Свойства поля
Должности	Код должности	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Должность	Текстовый	Размер поля: 20; обязательный ввод данных
	Оклад	Денежный	Обязательный ввод данных; условие на значение и сообщение об ошибке (табл. 4.5)
Ученые степени	Код степени	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Ученая степень	Текстовый	Размер поля: 20; обязательный ввод данных
	НадбавкаСтепень	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01
Стажи	Код стажа	Числовой	Первичный ключ; размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения не допускаются
	Стаж	Текстовый	Размер поля: 20; обязательный ввод данных
	НадбавкаСтаж	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01

Продолжение табл. 4.7

Таблица	Имя поля	Тип данных	Свойства поля
Уровни ГАК	Код ГАК	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Уровень ГАК	Текстовый	Размер поля: 20; обязательный ввод данных
	НадбавкаГАК	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01
Виды практик	Код практики	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Вид практики	Текстовый	Размер поля: 20; обязательный ввод данных в поле
	НадбавкаПрактика	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01
Руководство практикой	Код сотрудника	Числовой	Составной первичный ключ: размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются
	Код практики	Числовой	
	Число студентов	Числовой	Размер поля: целое, обязательный ввод данных; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 1

Продолжение табл. 4.7

Таблица	Имя поля	Тип данных	Свойства поля
Виды публикаций	Код публикации	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Вид публикации	Текстовый	Размер поля: 30; обязательный ввод данных в поле
	НадбавкаПубл	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01
Научные публикации	Код сотрудника	Числовой	Составной первичный ключ; размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются
	Код публикации	Числовой	
	Число публикаций	Числовой	Размер поля: целое, обязательный ввод данных; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 1
Виды НИР	Код НИР	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Вид НИР	Текстовый	Размер поля: 20; обязательный ввод данных в поле
	НадбавкаНИР	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01

Окончание табл. 4.7

Таблица	Имя поля	Тип данных	Свойства поля
Участие в НИР	Код сотрудника	Числовой	Составной первичный ключ; размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются
	Код НИР	Числовой	
	Число НИР	Числовой	Размер поля: целое, обязательный ввод данных; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 1.
Типы заявок	Код заявки	Счетчик	Первичный ключ, новые значения: последовательные, индекс с параметром: совпадения не допускаются
	Тип заявки	Текстовый	Размер поля: 30; обязательный ввод данных в поле
	НадбавкаЗаявка	Числовой	Размер: одинарное с плавающей точкой; формат: процентный; обязательный ввод данных; число десятичных знаков: 2; условие на значение и сообщение об ошибке (табл. 4.5); значение по умолчанию: 0,01
Заявки в Роспатент	Код сотрудника	Числовой	Составной первичный ключ; размер: длинное целое; обязательный ввод данных; индекс с параметром: совпадения допускаются
	Код заявки	Числовой	
Работа в ГАК	Код сотрудника	Числовой	Составной первичный ключ; длинное целое; обязательный ввод; индекс с параметром: совпадения допускаются
	Код ГАК	Числовой	

4.5. Разработка таблиц

Запустите СУБД MS Access. Создайте новую БД. Выполните команду **Сохранить базу данных как**. Выберите папку, согласитесь с предлагаемым по умолчанию типом файла и введите имя БД: **Заработная плата сотрудников ОУ**. Сохраните БД. Если появилась **Панель сообщений** с предупреждением системы безопасности об отключении запуска активного содержимого, то нажмите на этой панели кнопку **Включить содержимое**.

Начинаем разработку таблиц приложения в режиме конструктора таблиц. В этом режиме разработки действуют параметры по умолчанию (**Файл/Параметры/Конструкторы объектов**), которые следует просмотреть и, при желании изменить.

На вкладке **Создание** в группе **Таблицы** нажмите кнопку **Конструктор таблиц**. Разработку таблиц начнем с таблицы **Сотрудники**, имена и свойства полей которой определены на этапе проектирования (табл. 4.6). Введите имя первого поля: **Код сотрудника**; выберите тип данных поля из списка: **Счетчик**; введите описание: номер, автоматически присваиваемый новому сотруднику.

На вкладке **Конструктор таблиц** щелкните кнопку **Ключевое поле**. Просмотрите свойства поля в нижней области конструктора **Свойства**. Не изменяйте свойства, установленные по умолчанию. Свойство **Подпись** оставьте пустым, тогда в формах в качестве подписи будет использоваться имя поля.

Щелкните на вкладке **Таблица1** правой клавишей и выберите из контекстного меню команду **Сохранить**. Сохраните таблицу под именем **Сотрудники**. Далее выполняйте разработку полей таблицы **Сотрудники** в таком порядке: введите имя поля, выберите тип данных поля из списка, введите описание. В нижней области конструктора установите свойства

поля, определенные в процессе проектирования (табл. 4.6) Практически каждое свойство поля может быть выбрано из списка, некоторые (например, сообщение об ошибке, подпись) вводятся с клавиатуры. Для ввода формул в свойство **Условие на значение** (или **Правило проверки**) следует нажать кнопку **Построитель** на вкладке **Конструктор таблиц** или щелкнуть в правой части строки этого свойства. Откроется окно **Построитель выражений**.

Для поля **Дата приема** в построителе выражений следует ввести выражение: <= **Date ()**. Для ввода функции в области **Элементы выражения** из списка выбирается **Функции/Встроенные функции**; потом в области **Категории выражений** выбирается **Дата/Время**; и далее в области **Значения выражений** выбирается функция **Date**. Для вставки функции в выражение надо сделать двойной клик на имени. Для ввода знака «меньше или равно» в области **Элементы выражения** выбирается **Операторы**, далее в области **Категории выражений** выбирается **Сравнения** и в области **Значения выражений** выбирается нужный оператор. Для вставки оператора в выражение надо сделать двойной клик на операторе.

При создании выражений для проверки значений в других полях можно использовать арифметические операторы, логические, а также специальные операторы сравнения: **IN, BETWEEN, LIKE** [2].

Оператор **BETWEEN** определяет диапазон, значения которого должны уменьшаться. Для этого в выражение вводятся ключевое слово **BETWEEN**, начальное значение, ключевое слово **AND**, конечное значение.

Оператор **IN** определяет набор значений, в который данное значение может быть включено. Так, для проверки значений поля **Доля ставки** можно ввести выражение In (1,5;1,25;1;0,75;0,5;0,25).

Для проверки значений поля **Дата рождения** введите следующее выражение: (Year (Now ())-Year ([Дата рождения])>=25) And (Year (Now ())-Year ([Дата рождения])<=65).

Для проверки значений поля **Дата рождения** можно построить и другое, более сложное, но и более точное выражение, в котором использована встроенная функция **DateSerial (year, month, day)**, возвращающая числовое значение даты для указанного года, месяца и дня: (<DateSerial (Year (Date ())-25; Month (Date ()); Day (Date ()))) And (>DateSerial (Year (Date ())-65; Month (Date ()); Day (Date ()))).

Выполните разработку всех полей таблицы **Сотрудники** (табл. 4.6) и сохраните ее.

Откройте в режиме Конструктора таблицы **Окно свойств**. В строке **Условие на значение** введите:

[Дата приема]>= ([Дата рождения]+365*25) And [Дата приема]<= ([Дата рождения]+365*65).

В строке **Сообщение об ошибке** введите: Не выполняется требование к возрасту сотрудника на дату приема на работу: от 25 до 65 лет.

4.6. Создание индексов

Откройте в режиме конструктора таблицу **Сотрудники**. Нажмите кнопку **Индексы** на вкладке **Конструктор** и убедитесь в том, что ключевое поле в таблице проиндексировано автоматически, имя индекса: **PrimaryKey**.

Простой индекс для неключевого поля создается в режиме конструктора таблицы путем выбора значения **Да** в свойстве **Индексированное поле**. Установите это значение для поля **Фамилия**. Нажмите кнопку **Индексы** на вкладке **Конструктор** и убедитесь в том, что в таблице появился новый индекс. Если для свойства **Уникальный индекс** установить значение

Да, то в таблицу нельзя будет ввести две записи, совпадающие по полю **Фамилия**. В этом конкретном случае следует установить значение **Нет**, что обеспечит создание простого индекса с выбранным порядком сортировки записей по полю **Фамилия** и ускорит выполнение сортировок и запросов на выборку данных по полю **Фамилия**.

В целях освоения методики создания составного индекса вручную введите в окне **Индексы: Сотрудники** имя третьего индекса, например, **ФамИмяДР**. Рядом, в области **Имя поля**, выберите из списка поля таблицы: **Фамилия**, **Имя**, **Дата рождения**. В свойстве индекса **Уникальный индекс** установите значение **Да**. Такой индекс не позволит ввести в таблицу две записи, совпадающие по трем полям: **Фамилия**, **Имя**, **Дата рождения**. Сохраните таблицу **Сотрудники** с созданными индексами.

Далее переходите к разработке следующей таблицы приложения. Для создания составного первичного ключа, например, в таблице **Руководство практикой**, выделите два поля в конструкторе таблиц: **Код сотрудника**, **Код практики** и нажмите кнопку **Ключевое поле** на вкладке **Конструктор**.

Составной ключ из двух полей не позволит ввести записи в таблицу, полностью совпадающие по значениям в этих двух полях. Составной ключ реализуется СУБД автоматическим созданием индексной таблицы. Для того чтобы убедиться в этом, нажмите кнопку **Индексы** на вкладке **Конструктор.** В окне **Индексы: Руководство практикой** указано имя автоматически созданного индекса: **PrimaryKey** и свойства индекса **Уникальный индекс**.

После разработки всех таблиц приложения (табл. 4.7) просмотрите каждую таблицу с целью проверки правильности реализации принципов проектирования, убедитесь в отсутствии ошибок. Сохраните каждую таблицу и закройте все таблицы.

4.7. Создание элемента управления Поле со списком

Откройте таблицу **Сотрудники** в режиме конструктора. В этой таблице поля **Код должности**, **Код степени** являются внешними ключами, обеспечивающими связь с 1: М с главными таблицами.

Значения в поле внешнего ключа должны совпадать со значениями в поле первичного ключа, за счет этого обеспечивается целостность и непротиворечивость данных. Для ввода значений в поля внешних ключей пользователь должен знать, какие коды можно ввести в данное поле. Понятно, что пользователь при вводе и редактировании данных в таблицы не должен знать и помнить коды. Для этого разработчики БД обязаны проектировать дружелюбный пользовательский интерфейс, одним из признаков которого является использование выпадающих списков для выбора нужного значения поля. Это достигается использованием другого типа элемента управления: **Поле со списком**.

Выделите поле **Код должности**. В нижней части конструктора таблиц в области **Свойства поля** перейдите на вкладку **Подстановка**. На этой вкладке данных нет, лишь указано, что тип элемента управления: **Поле**.

На вкладке **Подстановка** тип элемента управления **Поле** может быть заменен на тип **Поле со списком**. Выберите из списка значение **Поле со списком**, чтобы увидеть ряд свойств этого элемента управления. На этой вкладке выбирается тип источника строк (таблица, запрос, список значений), создается (в конструкторе запросов) источник строк для списка, устанавливается число столбцов списка, ширина столбцов. Список может содержать один, два или более столбцов. Из них обязательно указывается **Присоединенный столбец**, точнее, порядковый номер столбца в источнике строк, в котором содержится то значение, которое должно быть сохранено

в поле таблицы. Остальные столбцы источника строк содержат значения, которые должен видеть пользователь, чтобы правильно выбрать нужный элемент из списка.

Для того чтобы вручную задать все свойства элемента управления **Поле со списком**, разработчик должен иметь некоторый опыт. Однако для начинающего разработчика в СУБД Access предусмотрен очень удобный интерфейс для автоматического заполнения всех свойств поля со списком.

Для того чтобы изменить тип элемента управления **Поле** на тип **Поле со списком** и задать свойства этого элемента, надо в конструкторе таблиц в верхней области щелкнуть нужное поле. В списке **Тип данных** выбрать значение **Мастер подстановок**. На первом шаге мастера выбирается **Тип источника строк** в выпадающем списке: таблица, запрос или список значений. Если в БД нет подходящей для выпадающего списка таблицы, то следует использовать список значений.

Для поля **Код должности** в таблице **Сотрудники** в списке **Тип данных** выберите значение **Мастер подстановок**. Создайте поле со списком, источником строк списка является таблица **Должности**. В список обязательно следует включить два поля: **Код должности** и **Должность**. Первое поле **Код должности** (это ключевое поле таблицы **Должности**) и будет **Присоединенным столбцом**. При этом его следует скрыть в списке, используя галочку (ширина столбца станет равна 0). Если сделать двойной клик на правой границе столбца **Должности**, то можно автоматически настроить его ширину по данным в этом поле. Поскольку данные еще не введены, то настроить ширину столбца можно будет позже в свойствах поля со списком. Установите сортировку значений в списке должностей, тогда пользователю будет удобно выбирать должность из списка, а в таблице **Сотрудники** будет сохраняться соответствующее значение кода должности из присоединенного столбца.

На вкладке **Подстановка** в области **Свойства поля** можно просмотреть свойства элемента управления **Поле со списком**, созданные мастером подстановок. Эти свойства можно редактировать вручную. Если возникла ошибка при разработке свойств, то лучше их полностью удалить (для этого следует выбрать элемент управления **Поле**), таблицу сохранить и затем снова запустить **Мастер подстановки**.

Выполните разработку всех полей со списками в таблице **Сотрудники** и оставшихся восьми таблицах приложения, которые приведены ниже (табл. 4.8). Здесь для каждого поля со списком указана таблица, которая должна использоваться для создания источника строк списка. Перечислены столбцы источника для поля подстановки, полужирным начертанием выделен столбец, по которому следует в мастере подстановок указать сортировку и использование в подписи. Курсивом выделен столбец, значения которого будут сохранены в таблице (присоединенный столбец).

Таблица 4.8

Сведения для создания полей со списками в таблицах

Таблица	Поле со списком	Таблица-источник строк списка	Столбцы источника для поля подстановки
Сотрудники	Код должности	Должности	*Код должности* **Должность**
Сотрудники	Код степени	Ученые степени	*Код степени* **Ученая степень**
Работа в ГАК	Код ГАК	Уровни ГАК	*Код ГАК* **Уровень ГАК**
Заявки в Роспатент	Код заявки	Типы заявок	*Код заявки* **Тип заявки**
Руководство практикой	Код практики	Виды практик	*Код практики* **Вид практики**

Окончание табл. 4.8

Таблица	Поле со списком	Таблица-источник строк списка	Столбцы источника для поля подстановки
Научные публикации	Код публикации	Виды публикаций	*Код публикации* **Вид публикации**
Участие в НИР	Код НИР	Виды НИР	*Код НИР* **Вид НИР**
Руководство практикой	Код сотрудника	Сотрудники	*Код сотрудника* **Фамилия** Имя Отчество
Научные публикации	Код сотрудника	Сотрудники	*Код сотрудника* **Фамилия** Имя
Участие в НИР	Код сотрудника	Сотрудники	*Код сотрудника* **Фамилия** Имя
Работа в ГАК	Код сотрудника	Сотрудники	*Код сотрудника* **Фамилия** Имя Отчество
Заявки в Роспатент	Код сотрудника	Сотрудники	*Код сотрудника* **Фамилия** Имя

В таблице **Сотрудники** есть еще одно поле **Доля ставки**, для которого следует создать элемент управления **Поле со списком**. Для создания этого элемента также используйте **Мастер подстановок**, на первом шаге которого переключатель следует поставить в положение: будет введен фиксированный набор значений. Укажите число столбцов списка: 1. Введите в строки значения, указанные в задании на проектирование: 1,5; 1,25; 1; 0,75; 0,5; 0,25. Если теперь изменить число стол-

бцов на 2, то появится возможность рядом с числовым значением списка ввести текстовый комментарий, например, полставки, четверть ставки, полторы ставки и т. д. Далее следует выбрать столбец, который должен быть сохранен в таблице **Сотрудники**. В примере это первый столбец. Далее согласитесь с подписью и поставьте флажок: ограничиться списком.

Сохраните каждую таблицу приложения и закройте все таблицы, поскольку следующим этапом разработки должно стать создание межтабличных связей.

4.8. Создание межтабличных связей

В БД закройте все таблицы. На вкладке **Работа с базами данных** нажмите кнопку **Схема данных.** В окне **Схема данных** из контекстного меню выберите команду: добавить таблицу. Выделите все таблицы приложения и нажмите кнопку **Добавить**. Используя указатель мыши, подберите размеры прямоугольников и их расположение в окне **Схемы данных** так, чтобы на схеме были видны наименования всех таблиц и все поля каждой таблицы.

В процессе проектирования приложения был определен перечень связанных таблиц приложения и указаны внешние ключи для создания каждой из 10 связей (табл. 4.4). Рассмотрим технологию создания связи на примере первой связи, указанной в первой таблице. Это связь между таблицами **Должности** и **Сотрудники** по полю **Код должности**.

Расположите эти таблицы рядом в окне **Схемы данных**. Выделите мышью поле **Код должности** в главной таблице **Должности** и перетащите его на поле **Код должности** в подчиненной таблице **Сотрудники**. После этого откроется окно **Изменение связей**, в котором СУБД уже будет определен тип отношения: «один-ко-многим». Вам остается поставить фла-

жок **Обеспечение целостности данных**. Это очень важный момент установки связи. Именно после этого СУБД включает все штатные встроенные средства по обеспечению целостности и непротиворечивости данных, которые рассмотрены в п. 1.1.5.

Выполните установку межтабличных связей в БД (табл. 4.4), используя описанную выше технологию. Если при установке связи СУБД определит тип отношения «один-к-одному», то следует эту связь отменить и проверить свойства поля внешнего ключа в подчиненной таблице (прежде всего, размер поля: длинное целое; свойства индекса: да, совпадения допускаются). Обратите внимание, что после установки всех связей в БД (табл. 4.4) таблица **надбавок за Стаж** остается несвязанной с другими таблицами. Это верно, поскольку данные этой таблицы будут использованы в запросе при вычислении надбавки за стаж каждому сотруднику.

Прежде чем завершить работу в окне **Схема данных** с сохранением макета, сделайте этот макет визуально понятным и красивым. Используя указатель мыши, расположите прямоугольники так, чтобы были видны все связи между таблицами и линии связей не пересекались.

4.9. Ввод данных в таблицы и тестирование приложения

Работу по созданию таблиц в БД приложения можно считать успешно завершенной, если выполнено тестирование таблиц путем ввода и редактирования записей. Введите записи в справочники БД, полностью соответствующие исходным данным (3.3) и условиям на значения полей (табл. 4.5).

Напомним, что под справочником понимается главная таблица в связи двух таблиц типа «один-ко-многим». Посколь-

ку все главные таблицы приложения известны (табл. 4.4), то заполнение справочников не должно вызвать проблем. Надо только следить за правописанием в записях таблиц. Откройте окно **Параметры Access** категория **Правописание**. Просмотрите и настройте при необходимости параметры автозамены и варианты автоматического исправления орфографии [11].

После ввода данных в главные таблицы следует выполнить тестирование правил проверки условий на значение полей (табл. 4.5). Для этого в таблицу введите запись, для которой СУБД должно выдать сообщение об ошибке. Ввод записи в таблицу можно отменить, дважды нажав клавишу **Esc**.

Таблица **надбавок за Стаж** не относится к главным таблицам, однако данные в нее следует внести. Пример данных в этой таблице приведен ниже (табл. 4.9).

После заполнения справочников введите записи в таблицу **Сотрудники**, содержащую сведения о сотрудниках ОУ и их профессиональной деятельности. Обратите внимание, насколько удобно работать в этой таблице, так как ранее была выполнена разработка полей со списками. Рекомендуется ввести 10–15 записей в таблицу **Сотрудники**, чтобы далее на этом множестве создавать запросы.

После ввода записей в таблицу **Сотрудники** следует ввести записи в связанные с ней таблицы: **Руководство практикой**, **Научные публикации**, **Участие в НИР, Работа в ГАК, Заявки в Роспатент**, расширяющие сведения о профессиональной деятельности сотрудников. Удобство работы в этих таблицах очевидно, ведь в каждой таблице для полей внешних ключей ранее были созданы поля со списками.

Известно, что СУБД автоматически контролирует появление повторяющихся записей, используя встроенные средства контроля уникальности первичных ключей.

Таблица 4.9

Пример надбавок за Стаж

Код стажа	Стаж	НадбавкаСтаж, %
0	< 5	0,00
1	>= 5, <10	1,00
2	>=10, <15	1,50
3	>= 15, <20	2,00
4	>= 20, <25	2,50
5	>=25, <30	3,00
6	>=30, <35	3,50
7	>= 35, <40	4,00
9	>= 40, <45	4,50
10	>= 45, <50	5,00

С целью проверки попытайтесь ввести две идентичные (по составному ключу из двух полей) записи в таблицу **Руководство практикой**. После появления сообщения СУБД об ошибке отмените ввод записи, дважды нажав клавишу **Esc**. Выполните аналогичное тестирование на ввод повторяющихся записей в таблицы **Научные публикации**, **Участие в НИР, Работа в ГАК, Заявки в Роспатент**.

Далее попытайтесь ввести в таблицу **Сотрудники** две записи, имеющие одинаковые значения в полях **Фамилия**, **Имя**, **Дата рождения**. Эти поля включены в составной индекс этой таблицы, созданный ранее вручную. Убедитесь в том, что СУБД также не позволяет ввести записи, не являющиеся уникальными по составному индексу. Далее попытайтесь ввести две записи, имеющие одинаковые значения в поле **Фамилия**. Поскольку этот индекс не является уникальным, то записи будут сохранены в БД.

5. Создание запросов

5.1. Интерфейс работы с запросами

Для работы с запросами необходимо перейти на вкладку **Создание** и использовать команды группы **Запросы**. Для создания запроса в режиме конструктора нажимается кнопка **Конструктор запросов**, после чего СУБД открывает окно **Добавление таблицы** с тремя вкладками. После добавления нужной для запроса таблицы (или таблиц) активизируется вкладка **Конструктор**, где присутствуют четыре группы команд: **Результаты**, **Тип запроса**, **Настройка запроса**, **Показать или скрыть**.

Окно конструктора состоит из двух областей: верхней и нижней. В каждой области действует свое контекстное меню. В верхней области выводятся вкладки открытых запросов и таблицы, на основе которых строится запрос. Запрос может строиться и на основе других запросов. Таблицы и запросы в верхней части конструктора выводятся с установленными в БД связями.

В нижней части конструктора выводятся поля виртуальной таблицы, которая создается в процессе создания запроса. Чтобы вывести поле из таблицы в запрос, его надо перетащить мышью в нижнюю область или дважды по нему щелкнуть (знак * выводит все поля таблицы в запрос). Чтобы удалить из запроса ненужное поле, следует выделить соответствующий столбец и удалить его.

В процессе создания запроса можно нажимать кнопку **Выполнить**, чтобы увидеть созданный на этом этапе запрос. После этого следует нажать кнопку **Конструктор**, чтобы снова переключиться в режим создания запроса. Для просмотра и редактирования запроса на языке SQL надо нажать кнопку **Режим/Режим SQL**.

Для каждого поля запроса в нижней части устанавливаются его свойства: сортировка, групповая операция, вывод на экран и условие отбора записей по этому полю. Условия отбора по полю можно формировать в построителе выражений (кнопка **Построитель** в группе **Настройка запросов**). Выражение можно набирать в области ввода, которая открывается клавишами <Shift> + F2.

В запросе можно задать несколько условий отбора по разным полям. Если условия отбора заданы в одной строке, то действует условие «И», если в разных строках — условие «ИЛИ».

В строке **Условие отбора** вместо конкретного выражения может быть задан параметр к запросу. Для этого в нужный столбец запроса в строку **Условие отбора** вводится сообщение пользователю, непременно в квадратных скобках. Если в строку **Условие отбора** поля **Фамилия** ввести: [Введите фамилию сотрудника], то СУБД выведет записи с указанной фамилией. Если в строку **Условие отбора** поля **Дата рождения** ввести: **Between** [Введите дату начальную] **And** [Введите дату конечную], то СУБД выведет записи, в которых дата рождения находится в указанном диапазоне, включая его границы.

Виртуальная таблица, которая формируется в процессе создания запроса, может содержать не только поля из физически существующих таблиц или поля других запросов, но и поля, вычисляемые в данном запросе. Надо отметить, что только с помощью запросов выполняются любые вычисления на множестве записей в таблицах БД.

Для создания вычисляемого поля необходимо ввести его имя в свободный столбец запроса. Причем это имя не должно совпа-

дать ни с одним именем поля, которое присутствует в верхней части конструктора запроса. После имени вычисляемого поля надо поставить знак двоеточие и далее открыть построитель, чтобы сформировать выражение для вычислений. В выражение в качестве операнда может быть включено любое поле из находящихся в верхней области конструктора запросов.

Проверить правильность выражения можно после закрытия построителя и выхода из данного поля. При наличии ошибки СУБД сразу выводит сообщение, позволяющее проанализировать и исправить ошибку. Для проверки логики вычислений можно запустить визуальный контроль, нажав кнопку **Выполнить**. Если условие отбора не выполняется, то запрос возвращает пустой набор записей. Чтобы получить непустой набор, следует изменить параметры вычисляемого выражения или его логику.

В группе **Показать или скрыть** есть кнопка **Итоги**, которая используется при создании итоговых запросов. Итоговые вычисления осуществляются только для группы записей, для одной записи они не имеют смысла. Поэтому для вычисления итогов в запросе необходимо предварительно записи сгруппировать по какому-либо полю. Кнопка **Итоги** отображает в конструкторе запроса строку **Групповая операция**. В строке **Групповая операция** функция **Группировка** выбирается в тех полях, по которым возможна группировка, а в остальных полях выбирается одна из нескольких итоговых функций.

Для использования условий отбора записей по какому-либо полю в строке **Групповая операция** для этого поля выбирается элемент **Условие**, при этом снимается флажок **Вывод на экран** для поля, в котором заполнена строка **Условие отбора**. Для вычисляемого поля в строке **Групповая операция** устанавливается элемент **Выражение**. Многие итоговые запросы создаются с использованием статистических функций (минимум, максимум, среднее и другие).

В заключении следует отметить, что запросы создаются в базе данных Access с помощью конструктора запросов, который настолько удобен и понятен, что позволяет даже начинающему разработчику создавать сложные запросы. Все запросы в БД хранятся на языке SQL. Для того чтобы просмотреть текст запроса на языке SQL, надо нажать кнопку **Режим/Режим SQL**. Просматривая инструкции языка SQL, при желании можно овладеть этим языком на хорошем уровне [2].

5.2. Технология создания запросов на выборку с вычисляемыми полями

Откройте БД «Заработная плата сотрудников ОУ». Создайте простейший запрос на выборку, цель которого состоит в получении персональных данных сотрудников учреждения: фамилия, имя, отчество, дата рождения. Для этого щелкните на вкладке **Создание** в группе **Запросы** и выберите команду **Конструктор запросов**. Выберите в окне **Добавление таблиц** таблицу **Сотрудники**. Внесите в нижнюю область запроса двойным щелчком по именам полей в таблице следующие поля: **Фамилия**, **Имя**, **Отчество**, **Дата рождения**. В строке **Сортировка** поля **Фамилия** выберите порядок сортировки: по возрастанию.

Сохраните запрос в БД под именем **Персоны**, выполнив команду **Сохранить** из контекстного меню вкладки **Запрос1.**

Нажмите кнопку **Выполнить** в группе **Результаты**, чтобы перейти в режим таблицы. Переключитесь в режим SQL и просмотрите текст инструкции. Оператор **SELECT** содержит слова **SELECT**, **FROM** и ключевое слово **ORDER BY**. За словом SELECT следуют сведения о том, какие именно поля необходимо включить в результирующий набор данных.

Для того чтобы задать условия отбора записей по дате рождения, вернитесь в режим конструктора. Введите в стро-

ку **Условие отбора** в поле **Дата рождения**: >= 01.01.1980. В строке **Сортировка** установите способ сортировки: по убыванию. Перейдите в режим таблицы, щелкнув по кнопке **Выполнить**. Если запрос вернул пустой набор записей, измените условие отбора, вернувшись в режим конструктора. Далее переключитесь в режим SQL и просмотрите текст инструкции. Теперь в текст инструкции добавлено ключевое слово **WHERE**, изменен и порядок сортировки.

Создадим запрос с параметром. Для этого введем в строку **Условие отбора** поля **Дата рождения** выражение:

Between [Введите дату начальную]
And [Введите дату конечную].

Запустите запрос на выполнение, вводя даты в окне ввода параметра. Просмотрите результаты запроса и его текст на языке SQL. Удалите в этом режиме из текста строку с ключевым словом **WHERE**. Выполните запрос, который теперь должен вернуть все записи таблицы **Сотрудники**.

Создадим в запросе **Персоны** вычисляемое поле, которое рассчитывает возраст сотрудника на текущую дату. Для этого в следующем пустом столбце бланка запроса в строку **Поле** введите имя вычисляемого поля: **Возраст** и после него двоеточие. Далее нажмите кнопку **Построитель** и введите выражение для определения возраста в годах[2]: Возраст:

[2] Арифметические операции с датами и временем можно выполнять обычным образом, поскольку Access так же как и Excel хранит даты и время как десятичные значения [10, 11]. Основной единицей измерения являются сутки. Они представляются последовательными десятичными значениями. Отсчет дат начинается с базовой даты: воскресенье 01 января 1900 года. При вводе даты и времени в поле Access сохраняет данные в виде десятичного значения (в формате с плавающей точкой), равного количеству дней, часов, минут и секунд между базовой и заданной датой. Базовая дата выбирается в Параметрах Excel. Время суток — это десятичная дробь, которая представляет часть суток между их началом (00:00 ночи) и заданным временем. Access назначает десятичные значения дням, часам, минутам и секундам, что позволяет выполнять сложные вычисления по формулам с датами и временем.

(Date ()-[Дата рождения])\365. В этом выражении использован оператор целочисленного деления на число дней в году.

Выполните запрос **Персоны**, убедитесь в том, что запрос работает и возвращает все записи из таблицы **Сотрудники**. Добавьте в бланк запроса поле **Дата приема**. Создайте вычисляемое поле **СтажСотр**. Для этого в построителе выражений введите: СтажСотр: (Date ()- [Дата приема])\365. Выполните запрос, который должен возвратить все записи таблицы **Сотрудники**.

Теперь приступим к созданию в этом запросе вычисляемого поля для вывода фамилии и инициалов сотрудника. Создайте вычисляемое поле с именем ФИО в последнем столбце бланка запроса. Создайте в построителе следующее выражение:

ФИО: [Сотрудники]! [Фамилия] & « « & UCase (Left ([Сотрудники]! [Имя];1)) & «. « & UCase (Left ([Сотрудники]! [Отчество];1)) & «.»

Выполните запрос **Персоны** и убедитесь в его правильной работе. Откройте запрос в режиме **SQL**. Просмотрите инструкцию. Вернитесь в режим конструктора запросов. Удалите из бланка запроса столбцы с полями: **Фамилия, Имя, Отчество**, поскольку эти поля сейчас не нужны в запросе: есть вычисляемое поле ФИО. обязательно добавьте в первый столбец бланка запроса поле **Код Сотрудника** из таблицы **Сотрудники**. Это поле будет использоваться в дальнейших запросах для вычисления заработной платы сотрудника.

Выделите столбец с полем **ФИО** и перетащите его в начало бланка запроса. Установите сортировку по полю **ФИО**. Обратите внимание, что для всех вычисляемых полей строка **Имя таблицы** в бланке запроса должна быть пустой. Флажок **Вывод на экран** должен быть установлен для всех полей. Выполните запрос **Персоны**, убедитесь в его правильной работе и сохраните запрос.

Создадим в этом же запросе **Персоны** поле для вычисления стажа сотрудника в сетке стажей (табл. 4.9). Можно счи-

тать, что это некоторый дискретный стаж сотрудника с шагом 5 лет. Введите в бланк запроса имя вычисляемого поля **Код стажа**, которое в точности должно совпадать с именем ключевого поля в таблице **Стажи** (табл. 4.7). Создайте в построителе выражений вычисляемое поле:

Код стажа: (((Date ()-[Сотрудники]! [Дата приема])\365)\5).

Выполните запрос **Персоны**, проверьте корректность его работы, сравнивая реальный стаж сотрудника (поле **СтажСотр**) с рассчитанным дискретным стажем (поле **Код стажа**). Сохраните запрос и закройте его.

Создадим запрос **НадбавкаЗаСтаж**. На вкладке **Создание** нажмите кнопку **Конструктор запросов**. Добавьте в запрос **Персоны** таблицу **Стажи**. Сохраните запрос с именем: **НадбавкаСтаж**. Добавьте в бланк запроса поля **Код сотрудника** из запроса **Персоны**, **Код стажа** и поле **НадбавкаСтаж** из таблицы **Стажи**. Выполните запрос. Вернитесь в режим конструктора. Щелкните в строке **Поле** третьего столбца и введите перед именем поля его алиас (новое имя в этом запросе): **ЗаСтаж %**. После этого поставьте двоеточие, далее следует имя поля из таблицы: **НадбавкаСтаж**. Выполните запрос и убедитесь, что третий столбец выводится с новым именем. Сохраните запрос **НадбавкаЗаСтаж** и закройте его.

Создадим запрос **НадбавкаЗаСтепень**. На вкладке **Создание** нажмите кнопку **Конструктор запросов**. Добавьте в запрос таблицы **Сотрудники**, **Ученые степени**. Сохраните запрос с именем: **НадбавкаЗаСтепень**. Добавьте в бланк запроса поле **Код сотрудника** из таблицы **Сотрудники, НадбавкаСтепень** из таблицы **Ученые степени**. Выполните запрос. Вернитесь в режим конструктора. Щелкните в строке **Поле** второго столбца и введите перед именем поля его алиас (новое имя в этом запросе): **ЗаСтепень %**. После этого поставьте двоеточие, далее следует имя поля из таблицы: **НадбавкаСтепень**. Выполните запрос и убедитесь, что второй столбец вы-

водится с новым именем. Сохраните запрос **НадбавкаЗаСтепень** и закройте его.

Создадим запрос, который выводит должность сотрудника, долю ставки по должности и оклад с учетом доли ставки, руб. На вкладке **Создание** нажмите кнопку **Конструктор запросов**. Добавьте в запрос таблицы **Сотрудники**, **Должности**. Сохраните запрос с именем: **ОкладСотрудника**. Добавьте в бланк запроса из таблицы **Сотрудники** поля **Код сотрудника, Доля ставки,** из таблицы **Должности** — поле **Должность**. В четвертом столбце бланка запроса сделайте вычисляемое поле **ОкладСотр**. Откройте **Построитель** и введите выражение: ОкладСотр: [Должности]! [Оклад]* [Сотрудники]! [Доля ставки].

Откройте окно **Свойства** вычисляемого поля и установите формат: денежный, число десятичных знаков — 2. Выполните запрос. Убедитесь, что возвращаются корректные данные. Сохраните и закройте запрос **ОкладСотрудника**.

5.3. Технология создания запроса на выборку с группировкой

Рассмотрим технологию создания запроса на выборку с группировкой на примере создания запроса, в котором будет вычисляться надбавка сотрудникам за руководство практиками студентов в процентах от оклада.

На вкладке **Создание** нажмите кнопку **Конструктор запросов**. Добавьте в запрос таблицы **Виды практик**, **Руководство практикой**. Сохраните запрос с именем: **НадбавкаЗаПракт**. Добавьте в бланк запроса поле **Код сотрудника** из таблицы **Руководство практикой**. Создайте во втором столбце запроса вычисляемое поле **НадбЗаПракт %**.

Откройте **Построитель** и введите выражение:

НадбЗаПракт %: Sum ([Руководство практикой]! [Число студентов]* [Виды практик]! [НадбавкаПрактика]).

На вкладке **Работа с запросами** в группе **Показать или скрыть** нажмите кнопку **Итоги**. В бланке запроса появится строка **Групповая операция**. Выберите в этой строке для поля **Код сотрудника** элемент **Группировка**, а для вычисляемого поля **НадбЗаПракт %** — элемент **Выражение**. Из контекстного меню этого поля выполните команду **Свойства**. В окне свойств установите формат поля — процентный, число десятичных знаков — 2. Закройте окно свойств.

Выполните запрос, просмотрите результат. Сохраните и закройте запрос.

По этой же технологии создаются запросы, требуемые по заданию:

- **НадбавкаЗаПубл**: Код сотрудника, **НадбЗаПубл %**.
- **НадбавкаЗаНИР**: Код сотрудника, **НадбЗаНИР %**.

Создайте эти два запроса самостоятельно.

Теперь создадим запрос для вычисления надбавок сотрудникам за подачу заявок в Роспатент также в процентах от оклада. На вкладке **Создание** нажмите кнопку **Конструктор запросов**. Добавьте в запрос таблицы **Типы заявок**, **Заявки в Роспатент**. Сохраните запрос с именем **НадбавкаЗаЗаяв**.

Добавьте в бланк запроса поле **Код сотрудника** из таблицы **Заявки в Роспатент**. Создайте во втором столбце запроса вычисляемое поле **НадбЗаЗаяв %**. Откройте **Построитель** и введите выражение:

НадбЗаЗаяв %: [Типы заявок]! [НадбавкаЗаявка].

На вкладке **Работа с запросами** в группе **Показать или скрыть** нажмите кнопку **Итоги**. В бланке запроса появится строка **Групповая операция**. Выберите в этой строке для поля **Код сотрудника** элемент **Группировка**, а для вычисляемого поля **НадбЗаЗаяв % —** элемент **Сумма**. Из контекст-

ного меню этого поля выполните команду **Свойства**. В окне свойств установите формат поля: процентный, число десятичных знаков — 2. Закройте окно свойств. Выполните запрос. Просмотрите результирующую таблицу. Сохраните запрос и закройте его. По этой же технологии создается еще один запрос, требуемый по заданию: **НадбавкаЗаГАК**: Код сотрудника, НадбЗаГАК %. Создайте этот запрос самостоятельно.

5.4. Создание запросов приложения

Рассмотренные выше технологии создания запросов в конструкторе запросов СУБД Access должны быть освоены на хорошем уровне, что позволит создать как названные выше запросы, так и два итоговых запроса, указанных в задании на разработку (п. 3.4): **Кадры** и **Зарплаты**.

Для создания запроса **Кадры** добавьте в качестве источника данных запросы **Персоны** и **ОкладСотрудника**. Для того чтобы объединить данные этих двух запросов, перетащите мышью поле **Код сотрудника** из запроса **Персоны** на поле **Код сотрудника** запроса **ОкладСотрудника**. Из контекстного меню линии связи выберите команду **Параметры объединения**. Установите переключатель во второе положение: объединение всех записей из запроса **Персоны** и только тех записей из запроса **ОкладСотрудника**, в которых связанные поля совпадают.

Выведите в бланк запроса поля следующие поля: **Код сотрудника**, **ФИО**, **Должность**, **Доля ставки**, **Дата рождения**, **Возраст**, **Дата приема**, **СтажСотр**.

Выполните запрос **Кадры** и убедитесь в корректности его работы. Сохраните запрос и закройте его.

Для создания запроса **Зарплаты** добавьте в качестве источника данных запросы: **Персоны, ОкладСотрудника, Над-**

бавкаЗаСтаж, НадбавкаЗаСтепень, НадбавкаЗаЗаявки, НадбавкаЗаГАК, НадбавкаЗаНИР, НадбавкаЗаПубл, НадбавкаЗаПракт.

Для того чтобы объединить данные всех запросов, надо установить параметры объединения для каждой пары запросов. Перетащите мышью поле **Код сотрудника** из запроса **Персоны** на поле **Код сотрудника** запроса **ОкладСотрудника**. Из контекстного меню линии связи выберите команду **Параметры объединения**. Установите переключатель во второе положение: объединение всех записей из запроса **Персоны** и только тех записей из запроса **ОкладСотрудника**, в которых связанные поля совпадают. Повторите эту процедуру еще семь раз, чтобы установить параметры объединения запроса **Персоны** с каждым из семи запросов, в которых вычислены надбавки.

Выведите в бланк запроса **Зарплаты** следующие поля: **Код сотрудника**, **ФИО**, **ОкладСотр**, **ЗаСтаж %**, **ЗаСтепень %, НадбЗаЗаяв %, НадбЗаГАК %**, **НадбЗаПракт %**, **НадбЗаПубл %, НадбЗаНИР %**. В последнем столбце бланка запроса создайте вычисляемое поле **ВсеНадб %**:

ВсеНадб %: ЗаСтаж % + ЗаСтепень % + НадбЗаЗаяв % + НадбЗаГАК % + НадбЗаПракт % + НадбЗаПубл % + НадбЗаНИР %.

В окне **Свойства** этого вычисляемого поля установите формат: процентный, число десятичных знаков: 2. Выполните запрос, убедитесь в корректности его результата. Вернитесь в режим конструктора и в последнем столбце бланка запроса создайте вычисляемое поле **Зарплата**: **ВсеНадб % * ОкладСотр.**

В окне **Свойства** этого вычисляемого поля установите формат — денежный, число десятичных знаков — 2. Выполните запрос **Зарплаты**, сохраните его и закройте.

При выполнении запроса **Зарплаты** может возникнуть ситуация, когда запрос выводит не все записи из таблицы

Сотрудники. Если сотрудник не руководил ни одним видом практики или не имеет ни одной научной публикации, ни одной заявки в Роспатент, не участвовал в НИР или не работал членом ГАК, то запись с его фамилией не появится в запросе **Зарплаты**.

Для того чтобы запрос **Зарплаты** работал корректно и выводил все записи из таблицы **Сотрудники**, следует некоторые поля этого запроса преобразовать в вычисляемые. В каждом вычисляемом поле надо использовать встроенную функцию **IIf** из категории **Управление**, которая возвращает одну из двух частей в зависимости от результата вычислений. Синтаксис этой функции: IIf (expr, truepart, falsepart). Для работы функции IIf потребуется встроенная функция **IsNull** из категории **Проверка**, которая возвращает логическое значение Null, если в выражении отсутствуют допустимые данные. Синтаксис функции: IsNull (expression).

Применительно к вычислению надбавок за практику в запросе **Зарплаты** следует выводить не поле **НадбЗаПракт %** из запроса **НадбавкаЗаПракт**, а вычисляемое поле:

ЗаПракт %: IIf (IsNull ([НадбавкаЗаПракт]! [НадбЗаПракт %]);0; [НадбавкаЗаПракт]! [НадбЗаПракт %]).

Аналогичные вычисляемые поля следует создать в запросе **Зарплаты** для вывода каждой из 5 надбавок, за исключением надбавки за ученую степень и стаж: **ЗаЗаяв %, ЗаГАК %**, **ЗаПракт %**, **ЗаПубл %, ЗаНИР %**.

Откройте запрос **Зарплаты** в режиме конструктора и выполните его редактирование, используя **Построитель**. Задайте выражения для вычислений в 6 полях **ЗаЗаяв %, ЗаГАК %**, **ЗаПракт %**, **ЗаПубл %, ЗаНИР %**, которые описаны выше. После этого следует выполнить редактирование вычисляемого поля ВсеНадб %:

ВсеНадб %: ЗаСтаж % + ЗаСтепень % + ЗаЗаяв % +
+ ЗаГАК % + ЗаПракт % + ЗаПубл % + ЗаНИР %.

В окне **Свойства** каждого вычисляемого поля установите формат — процентный, число десятичных знаков — 2. После внесения изменений в запрос выполните его и убедитесь в том, что запрос корректно выводит все записи из таблицы **Сотрудники** и все данные из связанных таблиц. Сохраните запрос. Откройте запрос в режиме SQL, чтобы посмотреть и оценить сложность запроса, созданного начинающим разработчиком приложений в СУБД Access.

6. Создание форм

6.1. Интерфейс работы с формами

Для создания формы в СУБД Access используются команды группы **Формы** на вкладке **Создание**. На этой вкладке есть несколько инструментов для быстрого создания форм, каждый из которых позволяет создать форму одним щелчком мыши. В режиме конструктора форм можно создать новую пустую форму или вносить изменения в существующую форму, добавлять настраиваемые элементы управления. В этом режиме активизируется группа вкладок **Работа с макетами форм**, в состав которой входят вкладки **Конструктор**, **Упорядочить** и **Формат**.

Если в области навигации выделить таблицу БД и нажать кнопку **Форма**, то автоматически будет создана форма для ввода данных одной записи в выделенную таблицу. В форму будут добавлены все поля таблицы, которая называется базовым источником данных. Новую форму можно сразу же начать использовать или при необходимости изменить ее в режиме макета или конструктора (кнопка **Режимы**). В режиме макета можно внести изменения в структуру формы при одновременном отображении данных. Например, можно настроить размер полей в соответствии с данными таблицы.

Если Access обнаруживает таблицу, связанную отношением «один-ко-многим» с таблицей, которая использовалась для

создания формы, то в форму добавляется подчиненная форма, в которой будут отображаться все записи связанной таблицы, относящиеся к текущей записи. Если существует несколько таблиц, связанных отношением «один-ко-многим» с таблицей, которая использовалась для создания формы в качестве базового источника данных, то Access не добавляет ни одной подчиненной формы в основную форму. Для связи данных в основной и подчиненной форме должно быть указано поле в строках **Основные поля**, **Подчиненные поля** на вкладке **Данные** в окне свойств подчиненной формы. Если необходимо выбрать поля для отображения на форме, то лучше воспользоваться мастером форм, с помощью которого легко создаются простые настраиваемые формы. В мастере можно определить группировку и сортировку данных и использовать поля из одной или нескольких таблиц или запросов (при условии предварительного указания связей между таблицами и запросами). В мастере форм можно получить различные результаты в зависимости от выбранных параметров. Поэтому можно запустить мастер несколько раз, экспериментируя с параметрами, пока не будет получена нужная форма.

Режим конструктора обеспечивает более подробное представление структуры формы. В нем отображаются разделы колонтитулов и данных формы. В этом режиме форма не выполняется, поэтому при внесении изменений невозможно просмотреть соответствующие данные. Однако некоторые задачи удобнее выполнять именно в режиме конструктора, например, добавление в форму элементов управления, изменение источников элемента управления, изменение размеров разделов формы.

С помощью элементов управления можно просматривать данные и работать с ними в приложении. Наиболее распространенным элементом является текстовое поле, кроме него используются кнопки, надписи, флажки, поля со списком (раскрывающиеся списки) и элементы управления подчиненной

формы. Весьма удобными для пользователя являются вычисляемые элементы управления — элементы управления, источником данных которых является выражение, а не поле. Для задания значения, которое должно содержаться в таком элементе управления, необходимо задать выражение, служащее источником данных элемента, используя построитель выражений.

В режиме конструктора можно изменить свойства формы, ее разделов и элементов управления с помощью окна свойств. Чтобы отобразить его, нажмите клавишу **F4.** Из области **Список полей** можно добавить в область данных формы поля из источника данных. Для отображения области **Список полей** на вкладке **Конструктор** в группе **Сервис** нажмите кнопку **Добавить поля**. Поля можно перетащить в форму непосредственно из области **Список полей**. Чтобы добавить одно поле, дважды щелкните его или перетащите его из области **Список полей** в тот раздел формы, где оно должно отображаться. Чтобы добавить сразу несколько полей, щелкните их, удерживая нажатой клавишу **CTRL**, и перетащите выбранные поля в форму.

Привязка элемента управления к полю осуществляется путем указания поля, из которого этот элемент управления получает данные. Чтобы создать элемент управления, связанный с выбранным полем, перетащите поле из области **Список полей** в форму. При двойном щелчке поля в области **Список полей** Access добавит в форму элемент управления соответствующего типа для этого поля. Можно также связать поле с элементом управления, введя имя поля в самом элементе управления (если объект открыт в режиме конструктора) или в качестве значения свойства **Данные** в окне свойств элемента управления. В окне свойств определяются характеристики элемента управления, например, имя, источник данных и формат.

При разработке формы рекомендуется переключаться из режима конструктора в режим формы, чтобы просмотреть и оценить результат работы.

6.2. Создание форм приложения

Разработку форм приложения удобнее всего начать с автоматического создания простых форм для ввода и редактирования данных в справочниках приложения. Перечень справочников приложения и имена форм приведены ниже (табл. 6.1). Для создания формы следует сначала выделить в области навигации таблицу — источник записей. Далее на вкладке **Создание** в группе **Формы** нажать кнопку **Форма** (можно выполнить и другую команду **Таблица** из списка **Другие формы**).

После создания формы можно выполнить ее редактирование, например, уменьшить ширину полей формы, изменить размер формы. Для одновременного одинакового изменения ширины полей формы их следует выделить при нажатой клавише **Ctrl**, далее, используя указатель мыши, переместить границу поля влево.

В форме **Должности** СУБД создает подчиненную форму, в которой выводятся записи из таблицы **Сотрудники**, связанные с текущей записью в форме. Для унификации форм и удобства работы только с записями справочника подчиненную форму надо выделить и удалить. Сохраните каждую форму и закройте.

Таблица 6.1

Перечень форм для справочников приложения

Форма	Источник записей
Должности	Должности
Ученые степени	Ученые степени
Уровни ГАК	Уровни ГАК
Типы заявок	Типы заявок
Виды практик	Виды практик
Виды публикаций	Виды публикаций
Виды НИР	Виды НИР
Стажи	Стажи

Теперь приступим к созданию всех форм (табл. 6.2), которые далее будут вставлены в основную форму **Сотрудники** как подчиненные, связанные с основной формой по полю **Код сотрудника**. Эти формы предназначены для добавления и редактирования данных о профессиональной деятельности сотрудника (руководстве практикой, работе в ГАК, участии в НИР, подаче заявок в Роспатент, научных публикациях). Эти формы также создаются автоматически СУБД, технология описана выше. Отметим главное достоинство созданных автоматически форм: наличие элемента управления **Поле со списком**, благодаря которому пользователь может легко редактировать записи и добавлять новые записи в таблицу. Для перемещения по записям и ввода новых записей в нижней части формы на панели **Запись** есть соответствующие кнопки.

Для каждой формы (табл. 6.2) установите ширину формы 8 см, предварительно следует уменьшить ширину полей в области данных и ширину надписи в области заголовка так, чтобы в форме не появилась полоса прокрутки по горизонтали (ее наличие создает некоторые неудобства пользователю). Изменение ширины полей и связанных с ними надписей надо выполнять в режиме макета формы.

Таблица 6.2

Перечень форм, подчиненных в форме Сотрудники

Форма	Источник записей — таблица
Работа в ГАК	Работа в ГАК
Заявки в Роспатент	Заявки в Роспатент
Руководство практикой	Руководство практикой
Научные публикации	Научные публикации
Участие в НИР	Участие в НИР

Теперь приступим к разработке основной формы **Сотрудники**, в которой пользователю удобно редактировать абсолют-

но все профессиональные данные о каждом сотруднике ОУ, согласно заданию на проектирование. В форме должны быть такие элементы управления, как подчиненные формы, поля со списком, вычисляемые поля ***(Возраст, Стаж)***. В этой форме следует обязательно использовать элемент управления — *поле со списком*. Например, для ввода данных о должности сотрудника должен использоваться список значений из таблицы **Должности**, в выпадающем списке коды первичных ключей должны быть скрыты, пользователь должен видеть в форме только текстовые значения (название должности).

Для создания формы **Сотрудники** выделите в области навигации ее источник данных — таблицу **Сотрудники**. Нажмите на вкладке **Создание** кнопку **Форма**. В режиме макета формы выделяем все поля в области данных и уменьшаем их ширину до 4 см. Переключаемся в режим конструктора. В группе **Элементы управления** проверяем, что кнопка **Использовать мастера** нажата. Выбираем элемент управления **Подчиненная форма/отчет**. Щелкаем в области данных формы **Сотрудники** в свободной верхней левой части рядом с полем **Код сотрудника**. Запускается мастер подчиненных форм. На его первом шаге устанавливаем переключатель в положение: имеющиеся формы — и выбираем форму **Работа в ГАК**. На следующем шаге задаем поля связи между главной и подчиненной формой: **Код сотрудника**. Далее соглашаемся с именем по умолчанию и нажимаем кнопку **Готово**. Теперь можно переключиться в режим формы, чтобы просмотреть форму, перемещаясь по записям в форме **Сотрудники**. Можно также добавить или редактировать запись в подчиненной форме, выбрав другой уровень ГАК.

Переключитесь в режим конструктора для продолжения вставки оставшихся трех подчиненных форм (табл. 6.2). Используя описанную выше технологию, вставьте рядом, справа от формы **ГАК**, форму **Заявки в Роспатент**, ниже вставьте еще 4 формы, по две в ряд. Сохраните форму **Сотрудники**.

Теперь осталось вставить в форму **Сотрудники,** в область данных, два вычисляемых поля, в которых соответственно выводится возраст и стаж сотрудника на текущую дату. В режиме конструктора увеличьте на 1 см высоту области данных для вставки в эту область вычисляемых полей.

В группе **Элементы управления** выберите элемент **Поле**, щелкните в нужном месте и протяните указатель мыши при нажатой левой клавише, чтобы указать размер поля. В окне свойств этого поля на вкладке **Данные** в строку **Данные** введите выражение для вычисления возраста, используя построитель выражений: = (Date ()-[Дата рождения])\365. Обратите внимание, что перед выражением следует обязательно поставить оператора равенства (=), иначе СУБД выдаст сообщение об ошибке.

На вкладке **Макет** укажите основной формат: число десятичных знаков 0, ширина 1 см, высота 0,5 см. Просмотрите результат в режиме формы. Переключитесь в режим конструктора, выделите надпись, связанную с вычисляемым полем, в окне ее свойств на вкладке **Макет**, в строке **Подпись** введите: **Возраст**. Используя приобретенный опыт, создайте рядом вычисляемое поле: = (Date ()- [Дата приема])\365 с надписью: **Стаж**.

После разработки формы **Сотрудники** выполните корректировку размеров элементов управления, полей и надписей, их взаимное расположение. Размер поля должен быть не больше, чем требуется для вывода данных в этом поле. Размер подчиненной формы должен быть подобран так, чтобы все данные в ней выводились без полос прокрутки. Постарайтесь сделать так, чтобы размер формы **Сотрудники** не превышал размера экрана, т. е. не было полос прокрутки. Можно также изменить формат формы. Однако надо понимать, что форма должна быть эргономичной, удобной для работы. Сохраните форму **Сотрудники** и закройте форму.

7. Создание отчетов

7.1. Интерфейс работы с отчетами

Для создания отчета в СУБД Access используются команды группы **Отчеты** на вкладке **Создание**. На этой вкладке есть кнопка **Отчет** для быстрого создания отчета на основе выделенного в области навигации объекта, который является источником данных (таблица или запрос). Простой отчет на основе данных таблицы или запроса создается одним щелчком мыши.

Команды **Конструктор отчета** и **Пустой отчет** создают пустые отчеты, в которые далее следует добавлять поля и элементы управления. Кнопка **Мастер отчетов** позволяет быстро создать простые настраиваемые отчеты. Запуск мастера наклеек — создать обычные или специальные наклейки.

Для работы с отчетами предназначены режимы: предварительный просмотр, представление, режим макета и режим конструктора. В режиме конструктора отчета и режиме макета активизируется группа вкладок: **Конструктор**, **Упорядочить**, **Формат**, **Параметры страницы**.

На вкладке **Конструктор** в группе **Группировка и итоги** находятся команды для задания уровней группировки и порядков сортировки данных в отчете. Группировка и сортировка упрощают чтение и анализ отчета, восприятие и понимание информации. Для каждой группы записей можно отобразить количество, сумму, среднее и другие операции

на группе записей. Как правило, группировка и сортировка выполняются по полю, выбранному в списке, но если требуется выполнить группировку или сортировку по вычисляемому значению, можно указать выражение.

Сортировка представляет собой процесс наложения порядка сортировки на записи в результатах запроса. Например, можно отсортировать записи по значению первичного ключа (или по другому набору значений в другом поле) по возрастанию или по убыванию либо отсортировать записи по одному или нескольким символам в указанном порядке, например, по алфавиту.

Для добавления группировки и сортировки в отчет необходимо выделить отчет в области навигации и выбрать в контекстном меню команду **Режим макета** или **Конструктор**. На вкладке **Конструктор** в группе **Группировка и итоги** нажать кнопку **Группировка**, после чего в области ниже отчета появится область **Группировка, сортировка и итоги**. Если кнопку **Группировка** отжать, то область скроется, а установленные уровни группировки сохранятся.

Чтобы добавить в отчет уровень группировки, выбирается команда **Добавить группировку**. Чтобы добавить в отчет порядок сортировки, выбирается команда **Добавить сортировку**. В области появится новый уровень группировки или порядок сортировки, а также список полей, доступных для группировки и сортировки данных для этого отчета.

Результаты вычислений по группе записей выводятся в отчете в области **Заголовок группы** или **Примечание группы**, которые создаются нажатием кнопки **Группировка**. Группировки в отчете могут быть вложенными, что позволяет создавать сложные итоговые отчеты. Поэтому в отчете появляются области **Заголовок группы, Примечание группы** с последующим указанием поля, которое используется в группировке.

Для выполнения вычислений на группе записей в областях **Заголовок группы** или **Примечание группы** необходимо

вставить элемент управления **Поле**, который выводит в отчете результат вычислений в соответствии с заданным выражением. Выражение лучше всего создавать в **Построителе**, обязательно начиная его с оператора равенства (=). Одно и то же вычисляемое поле можно вставлять в разные области группировки отчета, тогда результат вычислений будет сформирован по соответствующей группе записей.

Вычисляемое поле, которое находится в области **Заголовок отчета** или **Примечание отчета**, возвращает результат вычислений на множестве всех записей отчета.

7.2. Создание отчетов приложения

Выполним разработку отчетов, которые должны быть в создаваемом приложении (3.4). Создадим на основе запроса **Кадры** отчет **Кадровый состав**. В отчете должны быть выведены все поля запроса. В области **Заголовок отчета** выводится название отчета и текущая дата. В области **Примечание отчета** выводятся вычисляемые данные: всего сотрудников, средний возраст, средний стаж.

Выделите в области навигации запрос **Кадры** и нажмите кнопку **Отчет**. Откроется отчет в режиме макета. На вкладке **Параметры страницы** установите альбомную ориентацию.

На вкладке **Конструктор** нажмите кнопку **Группировка**. Добавьте сортировку по полю **ФИО**. Отожмите кнопку **Группировка**, чтобы закрыть область группировки в конструкторе отчета. Перейдите в режим конструктора, установите высоту области **Примечание отчета** примерно 2 см.

На вкладке **Конструктор** выберите элемент управления **Поле** и поместите его в области **Примечание отчета**. В окне свойств поля на вкладке **Данные** в строку **Данные** введите с помощью **Построителя** выражение для вычисления числа

сотрудников: =Count ([Код сотрудника]). Здесь используется встроенная функция **Count** из категории **Статистические**. Функция рассчитывает количество записей, возвращенных запросом. В связанную с вычисляемым полем надпись введите в окне свойств на вкладке **Макет** в строку **Подпись** текст: Число сотрудников. Переключитесь в режим макета, чтобы проверить результат. В этом режиме удобно настраивать ширину добавленных элементов управления и изменять их положение в отчете.

Используя описанную выше технологию, добавьте в область **Примечание отчета** поле, в котором вычисляется средний возраст сотрудников: =Avg ([Возраст]). Добавьте рядом еще одно поле для вычисления среднего стажа: =Avg ([СтажСотр]). Просмотрите отчет в режиме макета, чтобы убедиться в том, что он корректно отображает все данные. При необходимости настройте ширину полей и связанных с ними надписей. В области **Заголовок отчета** выполните редактирование надписи: **Кадровый состав**. Сохраните отчет под именем **Кадровый состав**.

Используя технологию создания отчета **Кадровый состав**, подробно описанную выше, создайте на основе запроса **Зарплаты** отчет **Заработная плата**, который выводит все поля запроса и следующие итоги по всему отчету: всего сотрудников, средняя зарплата, сумма зарплаты, руб.

Для освоения технологии создания отчетов с группировкой создайте с помощью **Мастера отчетов** отчет **Кадры**, с группировкой по полю **Должность** и вычислением среднего возраста сотрудников по каждой должности**.**

7.3. Создание и запуск макросов

Для освоения технологии работы с макросами решим следующую задачу. Предположим, что пользователю удобно

иметь в форме **Сотрудники** кнопку, при нажатии которой будет открываться отчет **Кадровый состав**, а разработчик приложения хотел бы при открытии этой же формы выводить некоторое информационное сообщение пользователю.

Для создания макроса перейдите на вкладку **Создание**, в группе **Макросы и код** выберите команду **Макрос**. Добавьте макрокоманду **Окно сообщения**. Введите в поле **Сообщение** текст по шаблону: «По вопросам работы формы и ее дизайну обращаться к разработчику по тел. ____».

Выберите тип сообщения: **Информационное**. Введите в поле **Заголовок** текст: **Контакты**. Из контекстного меню вкладки макроса выберите команду **Сохранить**. Сохраните макрос под именем **МакросИнф**.

Откройте форму **Сотрудники** в режиме конструктора. В окне свойств формы на вкладке **События** в строке **Открытие** выберите макрос **МакросИнф**. Проверьте запуск макроса при открытии формы в режиме формы.

Откройте форму **Сотрудники** в режиме конструктора. На вкладке **Конструктор** в группе **Элементы управления** нажмите элемент **Кнопка**. Проверьте, что нажата кнопка **Использовать мастера**. Щелкните в области заголовка формы **Сотрудники** в свободном месте и протяните указатель мыши, чтобы нарисовать кнопку. Откроется окно **Создание кнопок**. Выберите категорию **Работа с отчетом**, действие **Открыть отчет**, на следующем шаге выберите отчет **Кадровый состав**. Далее введите текст, который надо разместить на кнопке: **Отчет «Кадровый состав»**.

Перейдите в режим формы. Нажмите кнопку в форме и убедитесь в том, что отчет открывается. Сохраните форму **Сотрудники** и закройте все объекты приложения.

8. Подготовка приложения к работе

8.1. Создание кнопочной формы

Параметры запуска приложений MS Access устанавливаются в окне **Параметры Access** в категории команд **Текущая база данных**. Важным параметром запуска является кнопочная форма, которая служит в качестве точки входа в приложение MS Access и является своеобразным путеводителем по формам и отчетам БД. Для создания такой формы существует специальная надстройка, которая запускается командой **Диспетчер кнопочных форм**.

Из кнопочной формы может быть открыт отчет или форма для редактирования данных. Кроме того, элемент кнопочной формы может раскрывать новую страницу кнопочной формы, на которой располагаются соответствующие ей элементы формы. Кнопочная форма разрабатывается для того, чтобы обеспечить конечному пользователю комфортный режим работы с БД для выполнения всех функций в соответствии с должностными обязанностями.

Для создания кнопочной формы сначала необходимо выполнить настройку ленты. Для этого на вкладке **Файл** выбираете **Параметры**, далее — категорию **Настройка ленты**. В правой части окна выбираете вкладку **Создание** и нажимаете кнопку **Создать группу**, называете эту группу: Новая. В списке слева найдите команду **Диспетчер кно-**

почных форм и добавьте ее в группу **Новая** на вкладке **Создание**.

Запустите диспетчер кнопочных форм. Создайте в окне диспетчера форму **Кнопочная форма**, состоящую из 4 страниц: **Редактирование справочников**, **Сведения о сотрудниках**, **Просмотр отчетов, Выход из приложения**. Страница **Редактирование справочников** должна открывать страницу, на которой перечислены все 8 форм справочников приложения (табл. 6.1) и команда возврата в основное меню (команда **Перейти к кнопочной форме**).

Страница **Сведения о сотрудниках** должна открывать страницу, на которой находятся команды открытия форм: **Сотрудники, Кнопочная форма**. Страница **Просмотр отчетов** открывает следующую страницу для выбора элемента: отчет **Кадровый состав**, отчет **Заработная плата, Кнопочная форма.**

Каждая из страниц должна открывать следующую страницу для выбора конкретного объекта. Для этого в окне диспетчера создаете 4 страницы, вводите название каждой страницы. Далее выделяете последовательно каждую страницу, нажимаете кнопку **Изменить**. В окне **Изменение страницы кнопочной формы** нажимаете кнопку **Создать**. В окне **Изменение элемента кнопочной формы** вводите имя формы, выбираете команду **Открыть форму для изменения**, выбираете название формы. На каждой странице меню должна быть кнопка возврата на предыдущую страницу (команда **Перейти к кнопочной форме**).

После создания кнопочной формы в области навигации появятся таблица **Switchboard Items** и форма **Кнопочная форма**. Если возникает необходимость запустить диспетчер кнопочных форм повторно, например, чтобы исправить ошибку разработки, следует удалить из БД таблицу **Switchboard Items** и форму **Кнопочная форма**.

8.2. Настройка параметров приложения

Откройте вкладку **Файл**. Нажмите кнопку **Параметры** и выберите категорию **Текущая база данных**. Измените настройки в разделе **Параметры приложения**. В поле **Заголовок приложения** введите: **Расчет заработной платы**. В списке **Форма просмотра** выберите **Кнопочная форма**. Снимите флажок: **Включение режима макета**.

В разделе **Навигация** снимите флажок **Область навигации**, в разделе **Параметры ленты и панелей инструментов** снимите все флажки.

После настройки параметров для текущей БД при ее открытии теперь будут доступны только те объекты, которые указаны в кнопочной форме и в том режиме, который задан при ее разработке. Полный доступ к БД будет закрыт. Кнопочная форма будет автоматически запускаться при открытии БД. Область навигации будет скрыта.

Выполните на вкладке **Файл** команду **Сохранить**. Далее выполните команду **Закрыть базу данных**. Откройте БД. Убедитесь в том, что все команды на кнопочной форме открывают заданные объекты приложения.

Конечно, параметры, определяющие поведение при открытии БД, можно легко обойти и получить полный доступ к БД. Для этого следует заново настроить параметры на странице **Текущая база данных** диалогового окна **Параметры.** Чтобы обойти все установленные параметры запуска приложения, удерживайте нажатой клавишу **SHIFT** при открытии БД. В зависимости от параметров безопасности макросов, заданных для БД, при запуске может быть выведено одно или несколько предупреждений о безопасности. Клавишу **SHIFT** следует удерживать нажатой до тех пор, пока не закроются все предупреждения, иначе обойти параметры запуска не удастся.

8.3. Шифрование приложения с использованием пароля

В СУБД Access версии 2010 и выше не поддерживается защита на уровне пользователя для БД (см. п. 2). Однако при открытии БД из более ранней версии Access, имеющей защиту на уровне пользователя, в Access эти параметры будут продолжать действовать. Разрешения на уровне пользователей не защищают БД от злоумышленников и не предназначены для использования в качестве защитного барьера. Эту функцию следует использовать для повышения удобства работы с БД для надежных пользователей. Чтобы защитить данные, следует предоставить доступ к файлу БД и связанным с ней файлам безопасности на уровне пользователей только надежным пользователям, используя разрешения файловой системы Windows.

Чтобы не допустить несанкционированного использования БД СУБД Access, ее следует зашифровать, задав пароль. Если пароль для зашифрованной БД известен, ее можно расшифровать, а пароль удалить. Для шифрования следует использовать надежные пароли, представляющие собой сочетание прописных и строчных букв, цифр и символов. Пароли, не содержащие набор таких элементов, являются ненадежными. Рекомендуется использовать фразу-пароль, состоящую из 14 или более знаков. Очень важно запомнить пароль, иначе восстановить его и расшифровать БД будет невозможно.

Для шифрования созданного приложения его следует открыть в монопольном режиме. Для этого на вкладке **Файл** нажмите кнопку **Открыть**, выделите нужный файл. Нажмите стрелку рядом с кнопкой **Открыть** и выберите вариант **Монопольно**. Далее на вкладке **Файл** нажмите кнопку **Сведения** и выберите команду **Зашифровать паролем**. В окне диалога **Задание пароля базы данных** введите пароль в поле **Пароль**, повторите его в поле **Подтверждение** и нажмите кнопку **ОК**.

В процессе шифрования происходит перемешивание данных в таблицах, что исключает несанкционированный просмотр этих данных. Средство шифрования в Access сочетает в себе два улучшенных средства прежних версий — кодирование и пароли БД. При использовании пароля для шифрования БД все данные становятся нечитаемыми в других программах, и для того чтобы использовать эту БД, пользователи должны вводить пароль. Открытие зашифрованной БД выполняется точно так же, как и любой другой БД, только в открывшемся окне диалога необходимо ввести пароль. При открытии зашифрованной БД все пользователи имеют возможность просмотра всех ее объектов.

Вопросы для самопроверки

1. Поясните, что понимается под процессом информатизации?
2. Поясните, что такое информационная технология?
3. В чем состоит цель информационной технологии?
4. Что понимается под информационным ресурсом?
5. Какова значимость информационного ресурса для страны?
6. Поясните, в чем заключается глобальное значение информационных технологий?
7. Приведите определение базы данных.
8. Какие модели данных принято выделять при классификации баз данных?
9. Когда, кем и на основе какой теории была разработана реляционная модель данных?
10. Перечислите важнейшие понятия реляционной модели данных.
11. В чем достоинства реляционной модели данных?
12. Перечислите требования к данным реляционной таблицы.
13. В каком случае таблица называется отношением (relation)?
14. Что называется первичным ключом (primary key)?
15. Для чего используется составной первичный ключ (composite primary key)?
16. Что называется внешним ключом (foreign key)?

17. Перечислите требования к проектированию внешнего ключа в таблице.

18. Как называется таблица, содержащая внешний ключ?

19. Как называется таблица, содержащая первичный ключ, определяющий значения внешнего ключа?

20. Приведите правило ссылочной целостности относительно данных в поле первичного ключа.

21. Когда и как СУБД контролирует корректность значений внешних ключей?

22. В чем состоит особенность работы СУБД с ключевыми полями?

23. Какую роль играют индексы (или индексные таблицы), созданные в таблицах БД?

24. Как создается простой индекс в БД? Какова структура простого индекса?

25. Как создается составной индекс в БД? Какова структура составного индекса?

26. Что такое индекс (индексная таблица) в БД? Из чего состоит индекс?

27. Как создать составной индекс в таблице для сортировки данных по нескольким полям?

28. Как выполняется поиск данных в БД при наличии индекса?

29. Для чего используется свойство поля «Условие на значение»?

30. Для чего используется свойство таблицы «Условие на значение»?

31. Когда СУБД выдает сообщение об ошибке данных в одном поле?

32. Когда СУБД выдает сообщение об ошибке данных в полях таблицы?

33. С какой целью в реляционной БД устанавливаются связи между таблицами?

34. Приведите определение межтабличной связи «один-к-одному» (1:1) двух таблиц А и В.

35. Приведите определение межтабличной связи «один-ко-многим» (1: М) двух таблиц А и В.

36. Приведите определение межтабличной связи «многие-ко-многим» (М: М) двух таблиц А и В.

37. Как СУБД контролирует условие целостности для связи типа «один-ко-многим»?

38. Почему при установке связи «один-ко-многим» СУБД предлагает установить другой тип связи: «один-к-одному»?

39. В каком виде хранятся в памяти ЭВМ данные в поле типа Дата/время?

40. Поясните порядок создания элемента управления Поле со списком в подчиненной таблице.

41. Когда возникает свойство поля «Присоединенный столбец», и что оно означает?

42. Как в отчете создается группировка данных по полю?

43. Какие данные могут быть выведены в разделе Примечание отчета?

44. Как создается вычисляемое поле в форме?

45. Как устанавливается связь между основной и подчиненной формой?

46. Какие значения может принимать свойство поля «Индексированное поле»?

47. Что называется Схемой данных?

48. Для чего используется Схема данных?

49. Что означает важнейшее свойство БД, называемое целостностью?

50. Какими средствами СУБД поддерживает целостность БД?

51. Что такое ссылочная целостность БД?

52. Когда и с какой целью используется принцип нормализации?

53. Сформулируйте основной принцип проектирования реляционных БД?

54. Что такое Structured Query Language (SQL)?

55. Что означает такое свойство, как непроцедурность языка программирования?

56. В какой период времени и почему начался бурный рост популярности настольных СУБД?

57. Поясните особенности обработки данных в настольных СУБД.

58. В чем состоит отличие сетевых многопользовательских версий настольных СУБД от чисто настольных.

59. Перечислите недостатки сетевых многопользовательских версий настольных СУБД.

60. Поясните особенности обработки данных в сетевых многопользовательских версиях настольных СУБД.

61. Что Вам известно о настольной СУБД dBase?

62. Перечислите недостатки настольных СУБД.

63. Какой базовый принцип работы с данными реализован в серверных СУБД?

64. Какие основные функции выполняет сервер баз данных?

65. Какой вид имеет синтаксис оператора SELECT?

66. Какое предложение можно использовать для фильтрации результатов, возвращаемых оператором SELECT?

67. Какие существуют типы запросов на изменение?

68. Поясните порядок создания итогового запроса.

69. Почему в процессе развития информационных технологий обостряется проблем защиты информации?

70. Какие компоненты определяют понятие безопасность информации?

71. Что означает компонент «конфиденциальность» в понятии безопасность информации?

72. Что означает компонент «целостность» в понятии безопасность информации?

73. Что означает компонент «доступность» в понятии безопасность информации?

74. Перечислите составляющие субъектно-объектной модели безопасности.

75. Как реализуется шифрование приложения с использованием пароля?

76. С какой целью выполняется установка параметров запуска приложения?

Список библиографических ссылок

1. Острейковский В. А. Информатика : учебник. М. : Высшая школа, 2009. 512 с.

2. Грубер М. Введение в SQL. М. : Лори, 1996. 379 с.

3. Дейт К. Дж. Введение в системы баз данных. 8-е изд. СПб. : Вильямс, 2016. 1328 с.

4. Кириллов В. В. Введение в реляционные базы данных : учебное пособие. СПб. : БХВ-Петербург, 2009. 464 с.

5. Кузовкин А. В. Управление данными : учебник. М. : Академия, 2010. 256 с.

6. Кузнецов С. Д. Основы баз данных. 2-е изд. М. : Бином, 2007. 488 с.

7. Ладыженский Г. М. Системы управления базами данных — коротко о главном // СУБД. 1995. № 1–4.

8. Сеннов А. С. Access 2010. Учебный курс : учебное пособие. СПб. : Питер, 2010. 288 с.

9. Федоров А., Елманова Н. Введение в базы данных // КомпьютерПресс. 2000. № 4.

10. Поддержка по Office [Электронный ресурс]. Режим доступа: http://office.microsoft.com/ru-ru/ (дата обращения: 08.11.2017).

11. Поддержка по Access [Электронный ресурс]. Режим доступа: http://office.microsoft.com/ru-ru/access/?CTT=1 (дата обращения: 08.11.2017)

12. Дженнингс Р. Microsoft Access 97 в подлиннике. Наиболее полное руководство : в 2 т. Т. 1. СПб. : BHV-Санкт-Петербург, 1997. 604 с.

Оглавление

Учебное издание

Лысенко Тамара Михайловна

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЙ В НАСТОЛЬНОЙ РЕЛЯЦИОННОЙ СУБД

Редактор О. С. Смирнова
Верстка Е. В. Ровнушкиной

Подписано в печать 21.08.2018. Формат 60×84 1/16.
Бумага писчая. Цифровая печать. Усл. печ. л. 6,7.
Уч.-изд. л. 4,7. Тираж 50 экз. Заказ 267.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620083, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 358-93-06, 350-58-20, 350-90-13
Факс: 8 (343) 358-93-06
http://print.urfu.ru

ЛЫСЕНКО ТАМАРА МИХАЙЛОВНА

Доцент департамента радиоэлектроники и связи Уральского федерального университета, кандидат технических наук, почетный работник высшего профессионального образования РФ.

Область научных интересов — цифровая обработка радиолокационных изображений, информационные технологии управления учебным процессом.